CICE: the Los Alamos Sea Ice Model
Documentation and Software User's Manual

Elizabeth C. Hunke and William H. Lipscomb
T-3 Fluid Dynamics Group, Los Alamos National Laboratory
Los Alamos NM 87545

August 25, 2006

Contents

1 Introduction **2**

2 Coupling with other climate model components **4**
2.1 Atmosphere . . . 4
2.2 Ocean . . . 6

3 Model components **7**
3.1 Horizontal transport . . . 8
3.1.1 Reconstructing area and tracer fields . . . 9
3.1.2 Locating departure triangles . . . 11
3.1.3 Integrating fluxes . . . 16
3.1.4 Updating state variables . . . 17
3.2 Transport in thickness space . . . 18
3.3 Mechanical redistribution . . . 21
3.4 Dynamics . . . 23
3.5 Thermodynamics . . . 25
3.5.1 Thermodynamic surface forcing . . . 26
3.5.2 New temperatures . . . 28
3.5.3 Growth and melting . . . 32

4 Numerical implementation **34**
4.1 Directory structure . . . 35
4.2 Grid, boundary conditions and masks . . . 37
4.3 Initialization and coupling . . . 38
4.4 Choosing an appropriate time step . . . 38
4.5 Model output . . . 39
4.6 Execution procedures . . . 40

5 Troubleshooting **42**
5.1 Initial setup . . . 42
5.2 Slow execution . . . 42
5.3 Debugging hints . . . 42
5.4 Known bugs . . . 43

Acknowledgments and Copyright **43**

Table of namelist options **44**

Index of primary variables and parameters **46**

General Index **55**

Bibliography **57**

1 Introduction

The Los Alamos sea ice model (CICE) is the result of an effort to develop a computationally efficient sea ice component for a fully coupled atmosphere-ice-ocean-land global climate model. It was designed to be compatible with the Parallel Ocean Program (POP), an ocean circulation model developed at Los Alamos National Laboratory for use on massively parallel computers [32, 6, 7]. The current version of the model has been enhanced greatly through collaborations with members of the Community Climate System Model (CCSM) Polar Climate Working Group, based at the National Center for Atmospheric Research (NCAR).

CICE has several interacting components: a thermodynamic model that computes local growth rates of snow and ice due to vertical conductive, radiative and turbulent fluxes, along with snowfall; a model of ice dynamics, which predicts the velocity field of the ice pack based on a model of the material strength of the ice; a transport model that describes advection of the areal concentration, ice volumes and other state variables; and a ridging parameterization that transfers ice among thickness categories based on energetic balances and rates of strain. Additional routines prepare and execute data exchanges with an external "flux coupler," which then passes the data to other climate model components such as POP.

This model release is CICE version 3.14. It replaces CICE 3.1, which was released in February 2004. Although the model physics is similar to that of version 3.0, the code has changed substantially. The following are the major changes:

- The mechanical redistribution module, **ice_mechred.F**, has been modified to run more stably in rapidly deforming regions, using a smooth participation function instead of the "G^*" step function. A new distribution function agrees better with observations.
- Boundary updates in the dynamics module **ice_dyn_evp.F** were altered so that the code runs more efficiently. Code that explicitly flushes underflows to zero is now available (commented out).
- A new formula for computing thickness category boundaries is available.
- The **ice_therm_vertical.F** module now features the capability for simulating fresh or saline ice.
- Ice advection, ridging and dynamics can be subcycled under the thermodynamic time step.
- Some loops in **ice_transport_remap.F** are now more vector-friendly.
- Global arrays have been eliminated, except for allocatable work arrays available in **ice_work.F**.
- Options for other atmosphere and ocean forcing data sets were added in **ice_flux_in.F**.
- Coupling options have been generalized.
- Timers for non-MPI runs have been standardized to use the F90 intrinsic *system_clock*.

	Atmosphere		**Ocean**
	Provided by the flux coupler to the sea ice model		
$z_\circ$	Atmosphere level height	F_{frzmlt}	Freezing/melting potential
$\vec{U}_a$	Wind velocity	T_w	Sea surface temperature
Q_a	Specific humidity	S	Sea surface salinity
ρ_a	Air density	$\nabla H_\circ$	Sea surface slope
Θ_a	Air potential temperature	$\vec{U}_w$	Surface ocean currents
T_a	Air temperature		
$F_{sw\downarrow}$	Shortwave radiation		
$F_{L\downarrow}$	Incoming longwave radiation		
F_{rain}	Rainfall rate		
F_{snow}	Snowfall rate		
	Provided by the sea ice model to the flux coupler		
$\vec{\tau}_a$	Wind stress	$F_{sw\Downarrow}$	Penetrating shortwave
F_s	Sensible heat flux	F_{water}	Fresh water flux
F_l	Latent heat flux	F_{hnet}	Net heat flux to ocean
$F_{L\uparrow}$	Outgoing longwave	F_{salt}	Salt flux
F_{evap}	Evaporated water	$\vec{\tau}_w$	Ice-ocean stress
α	Surface albedo		
T_{sfc}	Surface temperature		
	a_i Ice fraction		
	T_a^{ref} 2 m reference temperature (diagnostic)		
	Q_a^{ref} 2 m reference humidity (diagnostic)		
	F_{swabs} Absorbed shortwave (diagnostic)		

Table 1: Data exchanged between the flux coupler and the sea ice model.

- Some history fields and namelist options have been added.
- Various minor bugs have been fixed.
- The $\langle 1^\circ \rangle$ "gx1" grid is available in **input_templates/**, along with an ice restart file.

Generally speaking, subroutine names are given in *italic* and file names are **boldface** in this document. Symbols used in the code are `typewritten`, while corresponding symbols in this document are in the $math$ font which, granted, looks a lot like italic. A comprehensive index, including a glossary of symbols with many of their values, appears at the end. The organization of this software distribution is described in Section 4.1; most files and subroutines referred to in this documentation are part of the CICE code found in **cice/source/**, unless otherwise noted.

After many years "CICE" has finally become an acronym, for "Community Ice CodE." Originally CICE was shorthand for "sea ice," an amused nod to people outside the field who when told that we study sea ice, think we're saying "C" ice and have no idea what the letter C has to do with ice. We still pronounce the name like that, but there has been a small grass-roots movement underway to alter the model name's pronunciation from "sea ice" to what an Italian might say, chē′-chā or "chee-chay." Others choose to say sīs (English, rhymes with "ice"), sēs (French, like "cease"), or shē-ī-sō͡o ("Shii-aisu," Japanese).

2 Coupling with other climate model components

The sea ice model exchanges information with the other model components via a flux coupler. We use a recent version of the CCSM Flux Coupler [19] from NCAR. The flux coupler was originally intended to gather state variables from the component models, compute fluxes at the model interfaces, and return these fluxes to the component models for use in the next integration period, maintaining conservation of momentum, heat and fresh water. However, several of these fluxes are now computed in the ice model itself and provided to the flux coupler for distribution to the other components, for two reasons. First, some of the fluxes depend strongly on the state of the ice, and vice versa, implying that an implicit, simultaneous determination of the ice state and the surface fluxes is necessary for consistency and stability. Second, given the various ice types in a single grid cell, it is more efficient for the ice model to determine the net ice characteristics of the grid cell and provide the resulting fluxes, rather than passing several values of the state variables for each cell. These considerations are explained in more detail below.

The fluxes and state variables passed between the sea ice model and the flux coupler are listed in Table 1. By convention, directional fluxes are positive downward.

The ice fraction a_i (`aice`)[1] is the total fractional ice coverage of a grid cell. That is, in each cell,

$$\begin{array}{rl} a_i = 0 & \text{if there is no ice} \\ a_i = 1 & \text{if there is no open water} \\ 0 < a_i < 1 & \text{if there is both ice and open water,} \end{array}$$

where a_i is the sum of fractional ice areas for each category of ice. The ice fraction is used by the flux coupler to merge fluxes from the ice model with fluxes from the other components. For example, the penetrating shortwave radiation flux, weighted by a_i, is combined with the net shortwave radiation flux through ice-free leads, weighted by $(1 - a_i)$, to obtain the net shortwave flux into the ocean over the entire grid cell. The flux coupler requires the fluxes to be divided by the total ice area so that the ice and land models are treated identically (land also may occupy less than 100% of an atmospheric grid cell). These fluxes are "per unit ice area" rather than "per unit grid cell area."

2.1 Atmosphere

The wind velocity, specific humidity, air density and potential temperature at the given level height $z_\circ$ are used to compute transfer coefficients used in formulas for the surface wind stress and turbulent heat fluxes $\vec{\tau}_a$, F_s, and F_l, as described below. Wind stress is arguably the primary forcing mechanism for the ice motion, although the ice–ocean stress, Coriolis force, and slope of the ocean surface are also important [36]. The sensible and latent heat fluxes, F_s and F_l, along with shortwave and longwave radiation, $F_{sw\downarrow}$, $F_{L\downarrow}$ and $F_{L\uparrow}$, are included in the flux balance that determines the ice or snow surface temperature. As described in Section 3.5, these fluxes depend nonlinearly on the ice surface temperature T_{sfc}; the balance equation is iterated until convergence, and the resulting fluxes and T_{sfc} are then passed to the flux coupler.

The snowfall precipitation rate (provided as liquid water equivalent and converted by the ice model to snow depth) also contributes to the heat and water mass budgets of the ice layer. Although melt ponds generally form on the ice surface in the Arctic and refreeze later in the fall, reducing the total amount of fresh water that reaches the ocean and altering the heat budget of the ice, we currently have no melt pond parameterization; rain and all melted snow end up in the ocean.

Wind stress and transfer coefficients for the turbulent heat fluxes are computed in subroutine *atmo_boundary_layer* following [19]. For clarity, the equations are reproduced here in the present notation.

[1] `Typewritten` equivalents used in the code are described in the index.

The wind stress and turbulent heat flux calculation accounts for both stable and unstable atmosphere-ice boundary layers. Define the "stability"

$$\Upsilon = \frac{\kappa g z_\circ}{u^{*2}} \left(\frac{\Theta^*}{\Theta_a \left(1 + 0.606 Q_a\right)} + \frac{Q^*}{1/0.606 + Q_a} \right),$$

where κ is the von Karman constant, g is gravitational acceleration, and u^*, Θ^* and Q^* are turbulent scales for velocity, temperature and humidity, respectively:

$$\begin{aligned} u^* &= c_u \left| \vec{U}_a \right| \\ \Theta^* &= c_\theta \left(\Theta_a - T_{sfc} \right) \\ Q^* &= c_q \left(Q_a - Q_{sfc} \right), \end{aligned} \tag{1}$$

where the wind speed has a minimum value of 1 m/s. We have ignored ice motion in u^*, and T_{sfc} and Q_{sfc} are the surface temperature and specific humidity, respectively. The latter is calculated by assuming a saturated surface temperature T_{sfc}, as described in Section 3.5.1.

The exchange coefficients c_u, c_θ and c_q are initialized as

$$\frac{\kappa}{\ln(z_{ref}/z_{ice})}$$

and updated during a short iteration, as they depend upon the turbulent scales. Here, z_{ref} is a reference height of 10 m and z_{ice} is the roughness length scale for the given sea ice category. Υ is constrained to have magnitude less than 10. Further, defining $\chi = (1 - 16\Upsilon)^{0.25}$ and $\chi \geq 1$, the "integrated flux profiles" for momentum and stability in the unstable ($\Upsilon < 0$) case are given by

$$\begin{aligned} \psi_m = &\quad 2\ln\left[0.5(1+\chi)\right] + \ln\left[0.5(1+\chi^2)\right] - 2\tan^{-1}\chi + \frac{\pi}{2}, \\ \psi_s = &\quad 2\ln\left[0.5(1+\chi^2)\right]. \end{aligned}$$

In a departure from the parameterization used in [19], we use profiles for the stable case following [18],

$$\psi_m = \psi_s = -\left[0.7\Upsilon + 0.75\left(\Upsilon - 14.3\right)\exp\left(-0.35\Upsilon\right) + 10.7\right].$$

The coefficients are then updated as

$$\begin{aligned} c_u' &= \frac{c_u}{1 + c_u\left(\lambda - \psi_m\right)/\kappa} \\ c_\theta' &= \frac{c_\theta}{1 + c_\theta\left(\lambda - \psi_s\right)/\kappa} \\ c_q' &= c_\theta' \end{aligned}$$

where $\lambda = \ln\left(z_\circ / z_{ref}\right)$. The first iteration ends with new turbulent scales from equations (1). After five iterations the latent and sensible heat flux coefficients are computed, along with the wind stress:

$$\begin{aligned} C_l &= \rho_a \left(L_{vap} + L_{ice}\right) u^* c_q \\ C_s &= \rho_a c_p u^* c_\theta^* + 1, \\ \vec{\tau}_a &= \frac{\rho_a u^{*2} \vec{U}_a}{|\vec{U}_a|}, \end{aligned}$$

where L_{vap} and L_{ice} are latent heats of vaporization and fusion, ρ_a is the density of air and c_p is its specific heat. Again following [18], we have added a constant to the sensible heat flux coefficient in order to allow some heat to pass between the atmosphere and the ice surface in stable, calm conditions.

The atmospheric reference temperature T_a^{ref} is computed from T_a and T_{sfc} using the coefficients c_u, c_θ and c_q. Although the sea ice model does not use this quantity, it is most convenient for the ice model to perform this calculation. The atmospheric reference temperature is returned to the flux coupler as a climate diagnostic. The same is true for the reference humidity, Q_a^{ref}.

Additional details about the latent and sensible heat fluxes and other quantities referred to here can be found in Section 3.5.1.

2.2 Ocean

New sea ice forms when the ocean temperature drops below its freezing temperature, $T_f = -\mu S$, where S is the seawater salinity and $\mu = 0.054$ is the ratio of the freezing temperature of brine to its salinity. The ocean model performs this calculation; if the freezing/melting potential F_{frzmlt} is positive, its value represents a certain amount of frazil ice that has formed in one or more layers of the ocean and floated to the surface. (The ocean model assumes that the amount of new ice implied by the freezing potential actually forms.) In general, this ice is added to the thinnest ice category. The new ice is grown in the open water area of the grid cell to a specified minimum thickness; if the open water area is nearly zero or if there is more new ice than will fit into the thinnest ice category, then the new ice is spread uniformly over the entire cell.

If F_{frzmlt} is negative, it is used to heat already existing ice from below. In particular, the sea surface temperature and salinity are used to compute an oceanic heat flux F_w ($|F_w| \leq |F_{frzmlt}|$) which is applied at the bottom of the ice. The portion of the melting potential actually used to melt ice is returned to the coupler in F_{hnet}. The ocean model adjusts its own heat budget with this quantity, assuming that the rest of the flux remained in the ocean.

In addition to runoff from rain and melted snow, the fresh water flux F_{water} includes ice meltwater from the top surface and water melted or frozen (a negative flux) at the bottom surface of the ice. This flux is computed as the net change of fresh water in the ice and snow volume over the coupling time step, excluding frazil ice formation and newly accumulated snow.

There is a flux of salt into the ocean under melting conditions, and a (negative) flux when sea water is freezing. However, melting sea ice ultimately freshens the top ocean layer, since the ocean is much more saline than the ice. The ice model passes the net flux of salt F_{salt} to the flux coupler, based on the net change in salt for ice in all categories. In the present configuration, `ice_ref_salinity` is used for computing the salt flux, although the ice salinity used in the thermodynamic calculation has differing values in the ice layers.

A fraction of the incoming shortwave $F_{sw\Downarrow}$ penetrates the snow and ice layers and passes into the ocean, as described in Section 3.5.1.

Many ice models compute the sea surface slope $\nabla H_\circ$ from geostrophic ocean currents provided by an ocean model or other data source. In our case, the sea surface height $H_\circ$ is a prognostic variable in POP—the flux coupler can provide the surface slope directly, rather than inferring it from the currents. (The option of computing it from the currents is provided in subroutine *evp_prep*.) The sea ice model uses the surface layer currents $\vec{U}_w$ to determine the stress between the ocean and the ice, and subsequently the ice velocity $\vec{u}$. This stress, relative to the ice,

$$\vec{\tau}_w \quad = \quad c_w \rho_w \left| \vec{U}_w - \vec{u} \right| \left[\left(\vec{U}_w - \vec{u} \right) \cos\theta + \hat{k} \times \left(\vec{U}_w - \vec{u} \right) \sin\theta \right]$$

is then passed to the flux coupler (relative to the ocean) for use by the ocean model. Here, θ is the turning angle between geostrophic and surface currents, c_w is the ocean drag coefficient, ρ_w is the density of seawater (`dragw` $= c_w \rho_w$), and $\hat{k}$ is the vertical unit vector. The turning angle is necessary if the top ocean model

layers are not able to resolve the Ekman spiral in the boundary layer. If the top layer is sufficiently thin compared to the typical depth of the Ekman spiral, then $\theta = 0$ is a good approximation. Here we assume that the top layer is thin enough.

3 Model components

The Arctic and Antarctic sea ice packs are mixtures of open water, thin first-year ice, thicker multiyear ice, and thick pressure ridges. The thermodynamic and dynamic properties of the ice pack depend on how much ice lies in each thickness range. Thus the basic problem in sea ice modeling is to describe the evolution of the ice thickness distribution (ITD) in time and space.

The fundamental equation solved by CICE is [38]:

$$\frac{\partial g}{\partial t} = -\nabla \cdot (g\mathbf{u}) - \frac{\partial}{\partial h}(fg) + \psi, \tag{2}$$

where $\mathbf{u}$ is the horizontal ice velocity, $\nabla = (\frac{\partial}{\partial x}, \frac{\partial}{\partial y})$, f is the rate of thermodynamic ice growth, ψ is a ridging redistribution function, and g is the ice thickness distribution function. We define $g(\mathbf{x}, h, t)\, dh$ as the fractional area covered by ice in the thickness range $(h, h + dh)$ at a given time and location.

Equation (2) is solved by partitioning the ice pack at each grid point into discrete thickness categories. The number of categories can be set by the user, with a default value $N_C = 5$. (Five categories, plus open water, are sufficient to simulate the annual cycles of ice thickness, ice strength, and surface fluxes [2, 21].) Each category n has lower thickness bound H_{n-1} and upper bound H_n. The lower bound of the thinnest ice category, H_0, is set to zero. The other boundaries are chosen with greater resolution for small h, since the properties of the ice pack are especially sensitive to the amount of thin ice [24]. The continuous function $g(h)$ is replaced by the discrete variable a_{in}, defined as the fractional area covered by ice in the thickness range (H_{n-1}, H_n). We denote the fractional area of open water by a_{i0}, giving $\sum_{n=0}^{N_C} a_{in} = 1$ by definition.

Category boundaries are computed in *init_itd* using one of two formulas. The old formula, from [22], gives lower boundaries (in meters) of (0.0, 0.64, 1.39 2.47, 4.57) for categories 1 to 5 when N_C = 5. A new formula has been introduced in CICE 3.14 in response to user requests for boundaries that are round numbers. This formula gives boundaries (0.0, 0.60, 1.40, 2.40, 3.60) for N_C = 5. The old formula (kcatbound = 0 in the namelist) is the default. Users may substitute their own preferred boundaries in *init_itd*.

In addition to the fractional ice area, a_{in}, we define the following state variables for each category n:

- v_{in}, the ice volume, equal to the product of a_{in} and the ice thickness h_{in}.
- v_{sn}, the snow volume, equal to the product of a_{in} and the snow thickness h_{sn}.
- e_{ink}, the internal ice energy in layer k, equal to the product of the ice layer volume, v_{in}/N_i, and the ice layer enthalpy, q_{ink}. Here N_i is the total number of ice layers, with a default value $N_i = 4$, and q_{ink} is the negative of the energy needed to melt a unit volume of ice and raise its temperature to 0°C; it is discussed in Section 3.5. (NOTE: In the current code, $e_i < 0$ and $q_i < 0$ with $e_i = v_i q_i$.)
- e_{sn}, the snow energy, equal to the product of v_{sn} and the snow enthalpy, q_{sn}. Currently there is just one snow layer, but future versions of CICE will allow multiple snow layers. (Similarly, $e_s < 0$ in the code.)
- T_{sfn}, the surface temperature.

Since the fractional area is unitless, the volume variables have units of meters (i.e., m^3 of ice or snow per m^2 of grid cell area), and the energy variables have units of J/m^2.

The three terms on the right-hand side of (2) describe three kinds of sea ice transport: (1) horizontal transport in (x, y) space; (2) transport in thickness space h due to thermodynamic growth and melting; and (3) transport in thickness space h due to ridging and other mechanical processes. We solve the equation by operator splitting in three stages, with two of the three terms on the right set to zero in each stage. We compute horizontal transport using the incremental remapping scheme of [5] as adapted for sea ice by [22]; this scheme is discussed in Section 3.1. Ice is transported in thickness space using the remapping scheme of [21], as described in Section 3.2. The mechanical redistribution scheme, based on [38], [30], [12], [9], and [23], is outlined in Section 3.3. To solve the horizontal transport and ridging equations, we need the ice velocity $\mathbf{u}$, and to compute transport in thickness space, we must know the the ice growth rate f in each thickness category. We use the elastic-viscous-plastic (EVP) ice dynamics scheme of [14], as modified by [13], [15] and [16], to find the velocity, as described in Section 3.4. Finally, we use the thermodynamic model of [3], discussed in Section 3.5, to compute f.

3.1 Horizontal transport

We wish to solve the continuity or transport equation,

$$\frac{\partial a_{in}}{\partial t} + \nabla \cdot (a_{in}\mathbf{u}) = 0, \tag{3}$$

for the fractional ice area in each thickness category n. Equation (3) describes the conservation of ice area under horizontal transport. It is obtained from (2) by discretizing g and neglecting the second and third terms on the right-hand side, which are treated separately (Sections 3.2 and 3.3).

There are similar conservation equations for ice volume, snow volume, ice energy, snow energy, and area-weighted surface temperature:

$$\frac{\partial v_{in}}{\partial t} + \nabla \cdot (v_{in}\mathbf{u}) = 0, \tag{4}$$

$$\frac{\partial v_{sn}}{\partial t} + \nabla \cdot (v_{sn}\mathbf{u}) = 0, \tag{5}$$

$$\frac{\partial e_{ink}}{\partial t} + \nabla \cdot (e_{ink}\mathbf{u}) = 0, \tag{6}$$

$$\frac{\partial e_{sn}}{\partial t} + \nabla \cdot (e_{sn}\mathbf{u}) = 0, \tag{7}$$

$$\frac{\partial \left(a_{in}T_{sfn}\right)}{\partial t} + \nabla \cdot (a_{in}T_{sfn}\mathbf{u}) = 0. \tag{8}$$

For simplicity, ice and snow are assumed to have constant densities, so that volume conservation is equivalent to mass conservation. We also transport the fractional area of open water, using (3) with $n = 0$. With $N_C = 5$ and $N_i = 4$ there are 46 transport equations to be solved.

Three transport schemes are available, upwind, MPDATA [34] and the incremental remapping scheme of [5] as modified for sea ice by [22]. Because so many fields are transported, the transport module is very expensive (close to half the total computer time) in runs using MPDATA. Although a cheaper first-order upwind scheme is available as an MPDATA option (see Table 4), we recommend using the incremental remapping method instead. This scheme has several desirable features:

- It conserves the quantity being transported (area, volume, or energy).
- It is non-oscillatory; that is, it does not create spurious ripples in the transported fields.
- It preserves tracer monotonicity. That is, it does not create new extrema in the thickness and enthalpy fields; the values at time $m + 1$ are bounded by the values at time m.

- It is second-order accurate in space and therefore is much less diffusive than first-order schemes. The accuracy may be reduced locally to first order to preserve monotonicity.

- It is efficient for large numbers of categories or tracers. Much of the work is geometrical and is performed only once per grid cell instead of being repeated for each quantity being transported.

The time step is limited by the requirement that trajectories projected backward from grid cell corners are confined to the four surrounding cells; this is what is meant by incremental remapping as opposed to general remapping. This requirement leads to a CFL-like condition,

$$\frac{\max|\mathbf{u}|\Delta t}{\Delta x} \leq 1.$$

For highly divergent velocity fields the maximum time step must be reduced by a factor of two to ensure that trajectories do not cross. However, ice velocity fields in climate models usually have small divergences per time step relative to the grid size.

The remapping algorithm can be summarized as follows:

1. Given mean values of the ice area and tracer fields in each grid cell, construct linear approximations of these fields. Limit the field gradients to preserve monotonicity.

2. Given ice velocities at grid cell corners, identify departure regions for the fluxes across each cell edge. Divide these departure regions into triangles and compute the coordinates of the triangle vertices.

3. Integrate these fields over the departure triangles to obtain the area, volume, and energy fluxes across each cell edge.

4. Transfer the fluxes across cell edges and update the state variables.

Since all scalar fields are transported by the same velocity field, step (2) is done only once per time step. The other three steps are repeated for each field in each thickness category. These steps are described below.

3.1.1 Reconstructing area and tracer fields

First, using the known values of the state variables, the ice area and tracer fields are reconstructed in each grid cell as linear functions of x and y. For each field we compute the value at the cell center (i.e., at the origin of a 2D Cartesian coordinate system defined for that grid cell), along with gradients in the x and y directions. The gradients are limited to preserve monotonicity. When integrated over a grid cell, the reconstructed fields must have mean values equal to the known state variables, denoted by $\bar{a}$ for fractional area, $\tilde{h}$ for thickness, and $\hat{q}$ for enthalpy. The mean values are not, in general, equal to the values at the cell center. For example, the mean ice area must equal the value at the centroid, which may not lie at the cell center.

Consider first the fractional ice area, the analog to fluid density ρ in [5]. For each thickness category we construct a field $a(\mathbf{r})$ whose mean is $\bar{a}$, where $\mathbf{r} = (x, y)$ is the position vector relative to the cell center. That is, we require

$$\int_A a\, dA = \bar{a}\, A, \tag{9}$$

where $A = \int_A dA$ is the grid cell area. Equation (9) is satisfied if $a(\mathbf{r})$ has the form

$$a(\mathbf{r}) = \bar{a} + \alpha_a \langle \nabla a \rangle \cdot (\mathbf{r} - \bar{\mathbf{r}}), \tag{10}$$

where $\langle \nabla a \rangle$ is a centered estimate of the area gradient within the cell, α_a is a limiting coefficient that enforces monotonicity, and $\bar{\mathbf{r}}$ is the cell centroid:

$$\bar{\mathbf{r}} = \frac{1}{A} \int_A \mathbf{r}\, dA.$$

It follows from (10) that the ice area at the cell center ($\mathbf{r} = 0$) is

$$a_c = \bar{a} - a_x \overline{x} - a_y \overline{y},$$

where $a_x = \alpha_a(\partial a/\partial x)$ and $a_y = \alpha_a(\partial a/\partial y)$ are the limited gradients in the x and y directions, respectively, and the components of $\bar{\mathbf{r}}$, $\overline{x} = \int_A x\, dA/A$ and $\overline{y} = \int_A y\, dA/A$, are evaluated using the triangle integration formulas described in Section 3.1.3. These means, along with higher-order means such as $\overline{x^2}$, $\overline{xy}$, and $\overline{y^2}$, are computed once and stored.

Next consider the ice and snow thickness and enthalpy fields. Thickness is analogous to the tracer concentration T in [5], but there is no analog in [5] to the enthalpy. The reconstructed ice or snow thickness $h(\mathbf{r})$ and enthalpy $q(\mathbf{r})$ must satisfy

$$\int_A a\, h\, dA = \bar{a}\, \tilde{h}\, A, \tag{11}$$

$$\int_A a\, h\, q\, dA = \bar{a}\, \tilde{h}\, \hat{q}\, A. \tag{12}$$

Equations (11) and (12) are satisfied when $h(\mathbf{r})$ and $q(\mathbf{r})$ are given by

$$h(\mathbf{r}) = \tilde{h} + \alpha_h \langle \nabla h \rangle \cdot (\mathbf{r} - \tilde{\mathbf{r}}), \tag{13}$$

$$q(\mathbf{r}) = \hat{q} + \alpha_q \langle \nabla q \rangle \cdot (\mathbf{r} - \hat{\mathbf{r}}), \tag{14}$$

where α_h and α_q are limiting coefficients, $\tilde{\mathbf{r}}$ is the center of ice area,

$$\tilde{\mathbf{r}} = \frac{1}{\bar{a}\, A} \int_A a\, \mathbf{r}\, dA,$$

and $\hat{\mathbf{r}}$ is the center of ice or snow volume,

$$\hat{\mathbf{r}} = \frac{1}{\bar{a}\, \tilde{h}\, A} \int_A a\, h\, \mathbf{r}\, dA.$$

Evaluating the integrals, we find that the components of $\tilde{\mathbf{r}}$ are

$$\tilde{x} = \frac{a_c \overline{x} + a_x \overline{x^2} + a_y \overline{xy}}{\bar{a}},$$

$$\tilde{y} = \frac{a_c \overline{y} + a_x \overline{xy} + a_y \overline{y^2}}{\bar{a}},$$

and the components of $\hat{\mathbf{r}}$ are

$$\hat{x} = \frac{c_1 \overline{x} + c_2 \overline{x^2} + c_3 \overline{xy} + c_4 \overline{x^3} + c_5 \overline{x^2 y} + c_6 \overline{xy^2}}{\bar{a}\, \tilde{h}},$$

$$\hat{y} = \frac{c_1 \overline{y} + c_2 \overline{xy} + c_3 \overline{y^2} + c_4 \overline{x^2 y} + c_5 \overline{xy^2} + c_6 \overline{y^3}}{\bar{a}\, \tilde{h}},$$

where

$$\begin{aligned} c_1 &\equiv a_c h_c, \\ c_2 &\equiv a_c h_x + a_x h_c, \\ c_3 &\equiv a_c h_y + a_y h_c, \\ c_4 &\equiv a_x h_x, \\ c_5 &\equiv a_x h_y + a_y h_x, \\ c_6 &\equiv a_y h_y. \end{aligned}$$

From (13) and (14), the thickness and enthalpy at the cell center are given by

$$h_c = \tilde{h} - h_x \tilde{x} - h_y \tilde{y},$$

$$q_c = \hat{q} - q_x \hat{x} - q_y \hat{y},$$

where h_x, h_y, q_x and q_y are the limited gradients of thickness and enthalpy. The surface temperature is treated the same way as ice or snow thickness, but it has no associated enthalpy.

We preserve monotonicity by van Leer limiting. If $\bar{\phi}(i,j)$ denotes the mean value of some field in grid cell (i,j), we first compute centered gradients of $\bar{\phi}$ in the x and y directions, then check whether these gradients give values of ϕ within cell (i,j) that lie outside the range of $\bar{\phi}$ in the cell and its eight neighbors. Let $\bar{\phi}_{\max}$ and $\bar{\phi}_{\min}$ be the maximum and minimum values of $\bar{\phi}$ over the cell and its neighbors, and let $\phi_{\max}$ and $\phi_{\min}$ be the maximum and minimum values of the reconstructed ϕ within the cell. Since the reconstruction is linear, $\phi_{\max}$ and $\phi_{\min}$ are located at cell corners. If $\phi_{\max} > \bar{\phi}_{\max}$ or $\phi_{\min} < \bar{\phi}_{\min}$, we multiply the unlimited gradient by $\alpha = \min(\alpha_{\max}, \alpha_{\min})$, where

$$\alpha_{\max} = (\bar{\phi}_{\max} - \bar{\phi})/(\phi_{\max} - \bar{\phi}),$$

$$\alpha_{\min} = (\bar{\phi}_{\min} - \bar{\phi})/(\phi_{\min} - \bar{\phi}).$$

Otherwise the gradient need not be limited.

3.1.2 Locating departure triangles

The method for locating departure triangles is discussed in detail by [5]. The basic idea is illustrated in Figure 1, which shows a shaded quadrilateral departure region whose contents are transported to the target or home grid cell, labeled H. The neighboring grid cells are labeled by compass directions: NW, N, NE, W, and E. The four vectors point along the velocity field at the cell corners, and the departure region is formed by joining the starting points of these vectors. Instead of integrating over the entire departure region, it is convenient to compute fluxes across cell edges. We identify departure regions for the north and east edges of each cell, which are also the south and west edges of neighboring cells. Consider the north edge of the home cell, across which there are fluxes from the neighboring NW and N cells. The contributing region from the NW cell is a triangle with vertices abc, and that from the N cell is a quadrilateral that can be divided into two triangles with vertices acd and ade. Focusing on triangle abc, we first determine the coordinates of vertices b and c relative to the cell corner (vertex a), using Euclidean geometry to find vertex c. Then we translate the three vertices to a coordinate system centered in the NW cell. This translation is needed in order to integrate fields (Section 3.1.3) in the coordinate system where they have been reconstructed (Section 3.1.1). Repeating this process for the north and east edges of each grid cell, we compute the vertices of all the departure triangles associated with each cell edge.

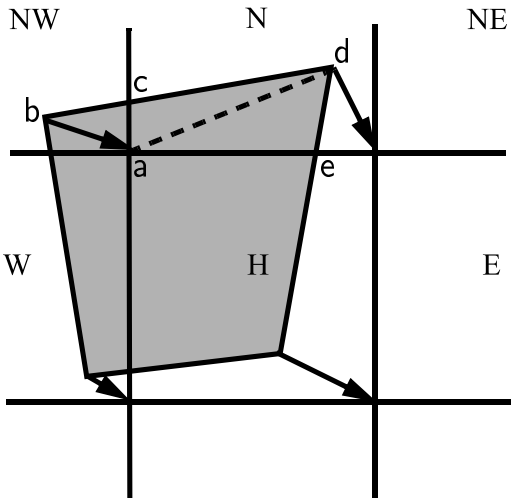

Figure 1: In incremental remapping, conserved quantities are remapped from the shaded departure region, a quadrilateral formed by connecting the backward trajectories from the four cell corners, to the grid cell labeled H. The region fluxed across the north edge of cell H consists of a triangle (abc) in the NW cell and a quadrilateral (two triangles, acd and ade) in the N cell.

Figure 2, reproduced from [5], shows all possible triangles that can contribute fluxes across the north edge of a grid cell. There are 20 triangles, which can be organized into five groups of four mutually exclusive triangles as shown in Table 2. In this table, (x_1, y_1) and (x_2, y_2) are the Cartesian coordinates of the departure points relative to the northwest and northeast cell corners, respectively. The departure points are joined by a straight line that intersects the west edge at $(0, y_a)$ relative to the northwest corner and intersects the east edge at $(0, y_b)$ relative to the northeast corner. This line intersects the north edge at $(x_a, 0)$ relative to the northwest corner and $(x_b, 0)$ relative to the northeast corner. From Cartesian geometry it can be shown that

$$\begin{aligned} y_a &= \frac{y_1\Delta x + (x_2y_1 - x_1y_2)}{\Delta x + x_2 - x_1}, \\ y_b &= \frac{y_2\Delta x + (x_2y_1 - x_1y_2)}{\Delta x + x_2 - x_1}, \\ x_a &= \frac{y_a\Delta x}{y_a - y_b}, \\ x_b &= \frac{y_b\Delta x}{y_a - y_b}, \end{aligned}$$

where Δx is the length of the north edge. The east cell triangles and selecting conditions are identical except for a rotation through 90 degrees.

This scheme was designed for rectangular grids. Grid cells in CICE actually lie on the surface of a sphere and must be projected onto a plane. Many such projections are possible. The projection used in CICE, illustrated in Figure 3, approximates spherical grid cells as quadrilaterals in the plane tangent to the sphere at a point inside the cell. The quadrilateral vertices are $(N/2, E/2)$, $(-N/2, W/2)$, $(-S/2, -W/2)$, and $(S/2, -E/2)$, where N, S, E, and W are the lengths of the cell edges on the spherical grid. The quadrilateral area, $(N + S)(E + W)/4$, is a good approximation to the true spherical area. However, cell edges in this projection are not orthogonal (i.e., they do not meet at right angles) as on the spherical grid. This means that when vectors are translated from cell corners to cell centers, we must take care that the

Triangle group	Triangle label	Selecting logical condition
1	NW	$y_a > 0$ and $y_1 \geq 0$ and $x_1 < 0$
	NW1	$y_a < 0$ and $y_1 \geq 0$ and $x_1 < 0$
	W	$y_a < 0$ and $y_1 < 0$ and $x_1 < 0$
	W2	$y_a > 0$ and $y_1 < 0$ and $x_1 < 0$
2	NE	$y_b > 0$ and $y_2 \geq 0$ and $x_2 > 0$
	NE1	$y_b < 0$ and $y_2 \geq 0$ and $x_2 > 0$
	E	$y_b < 0$ and $y_2 < 0$ and $x_2 > 0$
	E2	$y_b > 0$ and $y_2 < 0$ and $x_2 > 0$
3	W1	$y_a < 0$ and $y_1 \geq 0$ and $x_1 < 0$
	NW2	$y_a > 0$ and $y_1 < 0$ and $x_1 < 0$
	E1	$y_b < 0$ and $y_2 \geq 0$ and $x_2 > 0$
	NE2	$y_b > 0$ and $y_2 < 0$ and $x_2 > 0$
4	H1a	$y_a y_b \geq 0$ and $y_a + y_b < 0$
	N1a	$y_a y_b \geq 0$ and $y_a + y_b > 0$
	H1b	$y_a y_b < 0$ and $\tilde{y}_1 < 0$
	N1b	$y_a y_b < 0$ and $\tilde{y}_1 > 0$
5	H2a	$y_a y_b \geq 0$ and $y_a + y_b < 0$
	N2a	$y_a y_b \geq 0$ and $y_a + y_b > 0$
	H2b	$y_a y_b < 0$ and $\tilde{y}_2 < 0$
	N2b	$y_a y_b < 0$ and $\tilde{y}_2 > 0$

Table 2: Evaluation of contributions from the 20 triangles across the north cell edge. The coordinates x_1, x_2, y_1, y_2, y_a, and y_b are defined in the text. We define $\tilde{y}_1 = y_1$ if $x_1 > 0$, else $\tilde{y}_1 = y_a$. Similarly, $\tilde{y}_2 = y_2$ if $x_2 < 0$, else $\tilde{y}_2 = y_b$.

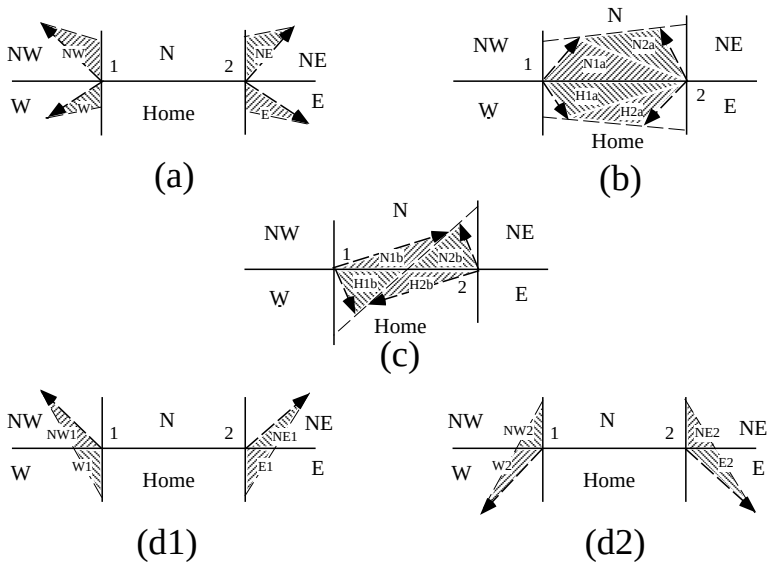

Figure 2: The 20 possible triangles that can contribute fluxes across the north edge of a grid cell.

departure points in the cell-center coordinate system lie inside the grid cell contributing the flux. Otherwise, monotonicity may be violated, because van Leer limiting does not apply outside the grid cell.

Figure 3 illustrates the difficulty. At the cell center we define orthogonal basis vectors $\hat{\imath}$ and $\hat{\jmath}$ that point toward the midpoints of the cell edges. Similarly, at each cell corner we define a coordinate system whose basis vectors, $\hat{\imath}'$ and $\hat{\jmath}'$, point along cell edges. The vectors $\hat{\imath}'$ and $\hat{\jmath}'$ are orthogonal in the cell-corner reference frame, but not when projected into the reference frame of the neighboring cell center. For this reason a simple transformation is needed to preserve monotonicity when vectors are translated from corners to centers. Consider a vector $(x'\hat{\imath}'+y'\hat{\jmath}')$ in the cell-corner basis. We make the approximation that this vector has the same coordinates when $\hat{\imath}'$ and $\hat{\jmath}'$ are non-orthogonal projections of the cell-corner basis vectors into the cell-center tangent plane, as in Figure 3. Then we transform from the $(\hat{\imath}', \hat{\jmath}')$ basis to the $(\hat{\imath}, \hat{\jmath})$ basis. In the cell-center coordinate system, $\hat{\imath}'$ is obtained by a rotation of $\hat{\imath}$ through an angle θ_N, where

$$\theta_N = \arctan\left(\frac{E-W}{2N}\right). \tag{15}$$

Similarly, $\hat{\jmath}'$ is obtained by a rotation of $\hat{\jmath}$ through θ_E, where

$$\theta_E = \arctan\left(\frac{S-N}{2E}\right). \tag{16}$$

Vectors are transformed between basis sets using

$$\begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} \cos\,\theta_N & -\sin\,\theta_E \\ \sin\,\theta_N & \cos\,\theta_E \end{pmatrix} \begin{pmatrix} x' \\ y' \end{pmatrix}, \tag{17}$$

which can be verified by inspection, alternately setting $x' = 0$ and $y' = 0$. Similar transformations are used at the other three cell corners. These transformations guarantee that the grid cell in which a given departure point is located does not change under a change in coordinate systems.

Most grids cells are nearly rectangular, unlike the distorted cell shown in Figure 3. On the 1° displaced-pole grid often used for CICE runs, the maximum angle in (15) and (16) is about 1°. Vector transformations

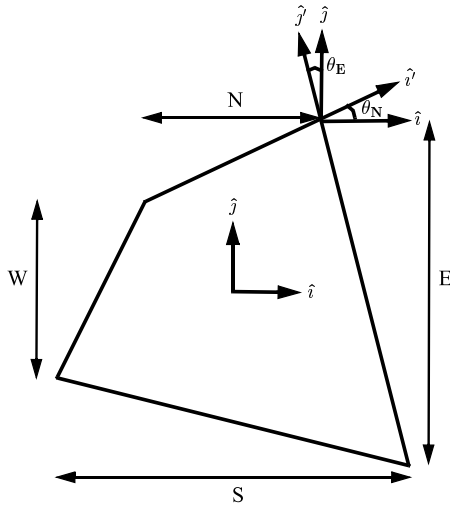

Figure 3: A grid cell on the surface of a sphere with unequal sides of length N, S, E, and W is approximated as a quadrilateral lying in the tangent plane at the cell center. The quadrilateral vertices are $(N/2, E/2)$, $(-N/2, W/2)$, $(-S/2, -W/2)$, and $(S/2, -E/2)$. The basis vectors $(\hat{i}', \hat{j}')$ at the northeast cell corner have been projected into the cell-center coordinate system and are different from the cell-center basis vectors $(\hat{\imath}, \hat{\jmath})$. The angles θ_N and θ_E relating the two bases are defined in the text.

may therefore be omitted on most grids with little loss of accuracy. We have retained them, however, because they ensure exact monotonicity at little added cost.

We made one other change in the scheme of [5] for locating triangles. In their paper, departure points are defined by projecting cell corner velocities directly backward. That is,

$$\mathbf{x}'_{\mathbf{D}} = -\mathbf{u}' \, \Delta t, \tag{18}$$

where $\mathbf{x}'_D$ is the location of the departure point relative to the cell corner and the primes denote vectors defined in the cell-corner basis. This approximation is only first-order accurate. Accuracy can be improved by correcting the velocity with a midpoint approximation before finding the departure point.

We first estimate the midpoint of the backward trajectory, $\mathbf{x}'_{\mathbf{M}} = \mathbf{x}'_{\mathbf{D}}/2$, then use an equation like (17) to transform $\mathbf{x}'_{\mathbf{M}}$ to the appropriate cell-center coordinate system. Next we use a bilinear interpolation to estimate the velocity at $\mathbf{x_M}$. In a square 2×2 grid cell with corners $\tilde{\mathbf{x}}_{\mathbf{1}} = (-1, -1)$, $\tilde{\mathbf{x}}_{\mathbf{2}} = (1, -1)$, $\tilde{\mathbf{x}}_{\mathbf{3}} = (1, 1)$, and $\tilde{\mathbf{x}}_{\mathbf{4}} = (-1, 1)$, the values of any scalar field ϕ can be matched at the cell corners with the following bilinear approximation:

$$\phi(\tilde{x}, \tilde{y}) = \frac{1}{4} \left[\phi_1 (\tilde{x} - 1)(\tilde{y} - 1) - \phi_2 (\tilde{x} + 1)(\tilde{y} - 1) + \phi_3 (\tilde{x} + 1)(\tilde{y} + 1) - \phi_4 (\tilde{x} - 1)(\tilde{y} + 1) \right], \tag{19}$$

where ϕ_1, ϕ_2, ϕ_3, and ϕ_4 are the corner values. To use (19) we must transform $\mathbf{x_M}$ from cell-center coordinates (x, y) into the simpler $(\tilde{x}, \tilde{y})$ coordinate system. Substituting x and y for ϕ in (19), we obtain

$$\begin{aligned} x(\tilde{x}, \tilde{y}) &= \frac{1}{4} \left[x_1 (\tilde{x} - 1)(\tilde{y} - 1) - x_2 (\tilde{x} + 1)(\tilde{y} - 1) + x_3 (\tilde{x} + 1)(\tilde{y} + 1) - x_4 (\tilde{x} - 1)(\tilde{y} + 1) \right], \\ y(\tilde{x}, \tilde{y}) &= \frac{1}{4} \left[y_1 (\tilde{x} - 1)(\tilde{y} - 1) - y_2 (\tilde{x} + 1)(\tilde{y} - 1) + y_3 (\tilde{x} + 1)(\tilde{y} + 1) - y_4 (\tilde{x} - 1)(\tilde{y} + 1) \right]. \end{aligned}$$

Recalling that the corner coordinates are $\mathbf{x_1} = (-S/2, -W/2)$, $\mathbf{x_2} = (S/2, -E/2)$ $\mathbf{x_3} = (N/2, E/2)$, and $\mathbf{x_4} = (-N/2, W/2)$, we can derive expressions for $\tilde{x}$ and $\tilde{y}$:

$$\tilde{x} = \frac{2x\Delta Y}{\Delta X \Delta Y + \delta X (2y - \tilde{x}\,\tilde{y}\,\delta Y)}, \tag{20}$$

$$\tilde{y} \;=\; \frac{2y\Delta X}{\Delta X\Delta Y + \delta Y(2x - \tilde{x}\,\tilde{y}\,\delta X)}, \tag{21}$$

where $\Delta X = (N+S)/2$, $\Delta Y = (E+W)/2$, $\delta X = (N-S)/2$, and $\delta Y = (E-W)/2$. These equations are nonlinear, since $\tilde{x}$ and $\tilde{y}$ appear on the right-hand side, but are easily iterated to convergence. Given the $(\tilde{x}, \tilde{y})$ coordinates of the midpoint, we apply (19) to the components of $\mathbf{u}$ at the cell corners to estimate the velocity at the midpoint. We transform the midpoint velocity back to cell-corner coordinates using the inverse of (17), then use the corrected velocity in (18) to find the departure point. With this correction, departure points for a velocity field varying linearly in space are nearly exact.

3.1.3 Integrating fluxes

Next, we integrate the reconstructed fields over the departure triangles to find the total fluxes of area, volume, and energy across each cell edge. Area fluxes are easy to compute since the area is linear in x and y. Given a triangle with vertices $\mathbf{x_i} = (x_i, y_i)$, $i \in \{1, 2, 3\}$, the triangle area is

$$A_T = \frac{1}{2}\left|(x_2 - x_1)(y_3 - y_1) - (y_2 - y_1)(x_3 - x_1)\right|. \tag{22}$$

The integral I_1 of any linear function $f(\mathbf{r})$ over a triangle is given by

$$I_1 = A_T f(\mathbf{x_0}), \tag{23}$$

where $\mathbf{x}_0 = (x_0, y_0)$ is the triangle midpoint,

$$\mathbf{x}_0 = \frac{1}{3}\sum_{i=1}^{3} \mathbf{x}_i. \tag{24}$$

To compute the area flux, we evaluate the area at the midpoint,

$$a(\mathbf{x}_0) = a_c + a_x x_0 + a_y y_0, \tag{25}$$

and multiply by A_T. By convention, northward and eastward fluxes are positive, while southward and westward fluxes are negative.

Equation (23) cannot be used for volume fluxes, because the reconstructed volumes are quadratic functions of position. (They are products of two linear functions, area and thickness.) The integral of a quadratic polynomial over a triangle requires function evaluations at three points,

$$I_2 = \frac{A_T}{3}\sum_{i=1}^{3} f\left(\mathbf{x}_i'\right), \tag{26}$$

where $\mathbf{x}_i' = (\mathbf{x}_0 + \mathbf{x}_i)/2$ are points lying halfway between the midpoint and the three vertices. [5] use this formula to compute fluxes of the product ρT, which is analogous to ice volume. Equation (26) does not work for ice and snow energies, which are cubic functions—products of area, thickness, and enthalpy. Integrals of a cubic polynomial over a triangle can be evaluated using a four-point formula [37]:

$$I_3 = A_T\left[-\frac{9}{16}f(\mathbf{x}_0) + \frac{25}{48}\sum_{i=1}^{3} f(\mathbf{x}_i'')\right] \tag{27}$$

where $\mathbf{x_i}'' = (3\mathbf{x}_0 + 2\mathbf{x}_i)/5$.

To evaluate functions at specific points, we must compute many products of the form $a(\mathbf{x})\,h(\mathbf{x})$ and $a(\mathbf{x})\,h(\mathbf{x})\,q(\mathbf{x})$, where each term in the product is the sum of a cell-center value and two displacement

terms. We can speed up the calculation by storing some sums that are used repeatedly. We first compute weighted ice areas at the four points of the integral formula (27):

$$a_0 = -\frac{9}{16}(a_c + a_x x_0 + a_y y_0),$$

$$a_i = \frac{25}{48}(a_c + a_x x_i + a_y y_i), i \in \{1, 2, 3\},$$

where we have dropped the double primes from the $\mathbf{x_i}$. We then define

$$\begin{aligned} \sigma_a &= \sum_{i=0}^{3} a_i, \\ \sigma_{ax} &= \sum_{i=0}^{3} a_i x_i, \\ \sigma_{ay} &= \sum_{i=0}^{3} a_i y_i, \end{aligned}$$

which we use to compute the volume fluxes:

$$\sigma_{ah} = \sigma_a h_c + \sigma_{ax} h_x + \sigma_{ay} h_y.$$

Note that σ_a, σ_{ax}, and σ_{ay} are used in three different flux calculations: ice volume, snow volume, and area-weighted surface temperature. Next we define

$$\begin{aligned} \sigma_{axx} &= \sum_{i=0}^{3} a_i x_i^2, \\ \sigma_{axy} &= \sum_{i=0}^{3} a_i x_i y_i, \\ \sigma_{ayy} &= \sum_{i=0}^{3} a_i y_i^2, \\ \sigma_{axh} &= \sigma_{ax} h_c + \sigma_{axx} h_x + \sigma_{axy} h_y, \\ \sigma_{ayh} &= \sigma_{ay} h_c + \sigma_{axy} h_x + \sigma_{ayy} h_y. \end{aligned}$$

The sums σ_{axh} and σ_{ayh} are computed separately for ice and snow, whereas the first three sums are independent of the material being transported. Each sum is used repeatedly if we have multiple enthalpy layers. From these sums we compute the energy fluxes:

$$\sigma_{ahq} = \sigma_{ah} q_c + \sigma_{axh} q_x + \sigma_{ayh} q_y,$$

thus completing the flux integrals for a given triangle. To compute the total flux across a cell edge we add the contributions from each triangle.

3.1.4 Updating state variables

Finally, we use these fluxes to compute new values of the state variables in each ice category and grid cell. The new fractional ice areas $a'_{in}(i,j)$ are given by

$$a'_{in}(i,j) = a_{in}(i,j) + \frac{F_{Ea}(i-1,j) - F_{Ea}(i,j) + F_{Na}(i,j-1) - F_{Na}(i,j)}{A(i,j)} \tag{28}$$

where $F_{Ea}(i,j)$ and $F_{Na}(i,j)$ are area fluxes across the east and north edges, respectively, of cell (i,j), and $A(i,j)$ is the grid cell area. All fluxes added to one cell are subtracted from a neighboring cell; thus (28) conserves total ice area.

The new ice volumes and energies are computed analogously. New thicknesses are given by the ratio of volume to area, and enthalpies by the ratio of energy to volume. Tracer monotonicity is ensured because

$$h' = \frac{\int_A a\,h\,dA}{\int_A a\,dA},$$

$$q' = \frac{\int_A a\,h\,q\,dA}{\int_A a\,h\,dA},$$

where h' and q' are the new-time thickness and enthalpy, given by integrating the old-time ice area, volume, and energy over a Lagrangian departure region with area A. That is, the new-time thickness and enthalpy are weighted averages over old-time values, with non-negative weights a and ah. Thus the new-time values must lie between the maximum and minimum of the old-time values.

3.2 Transport in thickness space

Next we solve the equation for ice transport in thickness space due to thermodynamic growth and melt,

$$\frac{\partial g}{\partial t} + \frac{\partial}{\partial h}(fg) = 0, \tag{29}$$

which is obtained from (2) by neglecting the first and third terms on the right-hand side. We use the remapping method of [21], in which thickness categories are represented as Lagrangian grid cells whose boundaries are projected forward in time. The thickness distribution function g is approximated as a linear function of h in each displaced category and is then remapped onto the original thickness categories. This method is numerically smooth (unlike schemes that treat $g(h)$ as a set of delta functions) and is not too diffusive. It can be viewed as a 1D simplification of the 2D incremental remapping scheme described above.

We first compute the displacement of category boundaries in thickness space. Assume that at time m the ice areas a_n^m and mean ice thicknesses h_n^m are known for each thickness category. (For now we omit the subscript i that distinguishes ice from snow.) We use a thermodynamic model (Section 3.5) to compute the new mean thicknesses h_n^{m+1} at time $m+1$. The time step must be small enough that trajectories do not cross; i.e., $h_n^{m+1} < h_{n+1}^{m+1}$ for each pair of adjacent categories. The growth rate at $h = h_n$ is given by $f_n = (h_n^{m+1} - h_n^m)/\Delta t$. By linear interpolation we estimate the growth rate F_n at the upper category boundary H_n:

$$F_n = f_n + \frac{f_{n+1} - f_n}{h_{n+1} - h_n}\,(H_n - h_n).$$

If a_n or $a_{n+1} = 0$, F_n is set to the growth rate in the nonzero category, and if $a_n = a_{n+1} = 0$, we set $F_n = 0$. The temporary displaced boundaries for $n = 1$ to $N-1$ are given by

$$H_n^* = H_n + F_n\,\Delta t.$$

The boundaries must not be displaced by more than one category to the left or right; that is, we require $H_{n-1} < H_n^* < H_{n+1}$. Without this requirement we would need to do a general remapping rather than an incremental remapping, at the cost of added complexity.

Next we construct $g(h)$ in the displaced thickness categories. The ice areas in the displaced categories are $a_n^{m+1} = a_n^m$, since area is conserved following the motion in thickness space (i.e., during vertical ice growth or melting). The new ice volumes are $v_n^{m+1} = (a_n h_n)^{m+1} = a_n^m h_n^{m+1}$. For conciseness, define

$H_L = H^*_{n-1}$ and $H_R = H^*_n$ and drop the time index $m+1$. We wish to construct a continuous function $g(h)$ within each category such that the total area and volume at time $m+1$ are a_n and v_n, respectively:

$$\int_{H_L}^{H_R} g\, dh = a_n, \tag{30}$$

$$\int_{H_L}^{H_R} h\, g\, dh = v_n. \tag{31}$$

The simplest polynomial that can satisfy both equations is a line. It is convenient to change coordinates, writing $g(\eta) = g_1\eta + g_0$, where $\eta = h - H_L$ and the coefficients g_0 and g_1 are to be determined. Then (30) and (31) can be written as

$$g_1\frac{\eta_R^2}{2} + g_0\eta_R = a_n,$$

$$g_1\frac{\eta_R^3}{3} + g_0\frac{\eta_R^2}{2} = a_n\eta_n,$$

where $\eta_R = H_R - H_L$ and $\eta_n = h_n - H_L$. These equations have the solution

$$g_0 = \frac{6a_n}{\eta_R^2}\left(\frac{2\eta_R}{3} - \eta_n\right), \tag{32}$$

$$g_1 = \frac{12a_n}{\eta_R^3}\left(\eta_n - \frac{\eta_R}{2}\right). \tag{33}$$

Since g is linear, its maximum and minimum values lie at the boundaries, $\eta = 0$ and η_R:

$$g(0) = \frac{6a_n}{\eta_R^2}\left(\frac{2\eta_R}{3} - \eta_n\right) = g_0, \tag{34}$$

$$g(\eta_R) = \frac{6a_n}{\eta_R^2}\left(\eta_n - \frac{\eta_R}{3}\right). \tag{35}$$

Equation (34) implies that $g(0) < 0$ when $\eta_n > 2\eta_R/3$, i.e., when h_n lies in the right third of the thickness range (H_L, H_R). Similarly, (35) implies that $g(\eta_R) < 0$ when $\eta_n < \eta_R/3$, i.e., when h_n is in the left third of the range. Since negative values of g are unphysical, a different solution is needed when h_n lies outside the central third of the thickness range. If h_n is in the left third of the range, we define a cutoff thickness, $H_C = 3h_n - 2H_L$, and set $g = 0$ between H_C and H_R. Equations (32) and (33) are then valid with η_R redefined as $H_C - H_L$. And if h_n is in the right third of the range, we define $H_C = 3h_n - 2H_R$ and set $g = 0$ between H_L and H_C. In this case, (32) and (33) apply with $\eta_R = H_R - H_C$ and $\eta_n = h_n - H_C$.

Figure 4 illustrates the linear reconstruction of g for the simple cases $H_L = 0$, $H_R = 1$, $a_n = 1$, and $h_n = 0.2$, 0.4, 0.6, and 0.8. Note that g slopes downward ($g_1 < 0$) when h_n is less than the midpoint thickness, $(H_L + H_R)/2 = 1/2$, and upward when h_n exceeds the midpoint thickness. For $h_n = 0.2$ and 0.8, $g = 0$ over part of the range.

Finally, we remap the thickness distribution to the original boundaries by transferring area and volume between categories. We compute the ice area Δa_n and volume Δv_n between each original boundary H_n and displaced boundary H^*_n. If $H^*_n > H_n$, ice moves from category n to $n+1$. The area and volume transferred are

$$\Delta a_n = \int_{H_n}^{H^*_n} g\, dh, \tag{36}$$

$$\Delta v_n = \int_{H_n}^{H^*_n} h\, g\, dh. \tag{37}$$

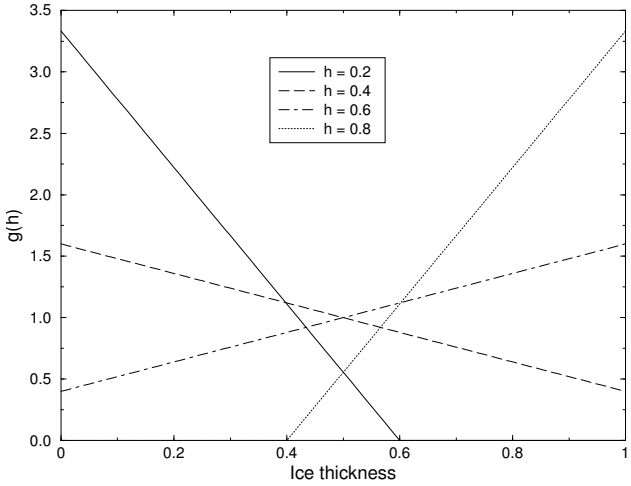

Figure 4: Linear approximation of the thickness distribution function $g(h)$ for an ice category with left boundary $H_L = 0$, right boundary $H_R = 1$, fractional area $a_n = 1$, and mean ice thickness $h_n = 0.2$, 0.4, 0.6, and 0.8.

If $H_n^* < H_N$, ice area and volume are transferred from category $n+1$ to n using (36) and (37) with the limits of integration reversed. To evaluate the integrals we change coordinates from h to $\eta = h - H_L$, where H_L is the left limit of the range over which $g > 0$, and write $g(\eta)$ using (32) and (33). In this way we obtain the new areas a_n and volumes v_n between the original boundaries H_{n-1} and H_n in each category. The new thicknesses, $h_n = v_n/a_n$, are guaranteed to lie in the range (H_{n-1}, H_n). If $g = 0$ in the part of a category that is remapped to a neighboring category, no ice is transferred.

Other conserved quantities are transferred in proportion to the ice volume Δv_{in}. (We now use the subscripts i and s to distinguish ice from snow.) For example, the transferred snow volume is $\Delta v_{sn} = v_{sn}(\Delta v_{in}/v_{in})$, and the transferred ice energy in layer k is $\Delta e_{ink} = e_{ink}(\Delta v_{in}/v_{in})$.

The left and right boundaries of the domain require special treatment. If ice is growing in open water at a rate F_0, the left boundary H_0 is shifted to the right by $F_0 \Delta t$ before g is constructed in category 1, then reset to zero after the remapping is complete. New ice is then added to the grid cell, conserving area, volume, and energy. If ice cannot grow in open water (because the ocean is too warm or the net surface energy flux is downward), H_0 is fixed at zero, and the growth rate at the left boundary is estimated as $F_0 = f_1$. If $F_0 < 0$, the area of ice thinner than $\Delta h_0 = -F_0 \Delta t$ is added to the open water area a_0, leaving the ice and snow volume and energy unchanged. The area of new open water is

$$\Delta a_0 = \int_0^{\Delta h_0} g \, dh.$$

The right boundary H_N is not fixed but varies with h_N, the mean ice thickness in the thickest category. Given h_N, we set $H_N = 3h_N - 2H_{N-1}$, which ensures that $g(h) > 0$ for $H_{N-1} < h < H_N$ and $g(h) = 0$ for $h \geq H_N$. No ice crosses the right boundary.

If the ice growth or melt rates in a given grid cell are too large, the thickness remapping scheme will not work. Instead, the thickness categories in that grid cell are treated as delta functions following [2], and categories outside their prescribed boundaries are merged with neighboring categories as needed. For time steps of less than a day and category thickness ranges of 10 cm or more, this simplification is needed rarely, if ever.

3.3 Mechanical redistribution

The last term on the right-hand side of (2) is ψ, which describes the redistribution of ice in thickness space due to ridging and other mechanical processes. The mechanical redistribution scheme in CICE is based on [38], [30], [12], [9], and [23]. This scheme converts thinner ice to thicker ice and is applied after horizontal transport. When the ice is converging, enough ice ridges to ensure that the ice area does not exceed the grid cell area.

First we specify the participation function: the thickness distribution $a_P(h) = b(h)\,g(h)$ of the ice participating in ridging. (We use "ridging" as shorthand for all forms of mechanical redistribution, including rafting.) The weighting function $b(h)$ favors ridging of thin ice and closing of open water in preference to ridging of thicker ice. There are two options for the form of $b(h)$. If $\texttt{krdg}_{\texttt{partic}} = \texttt{0}$ in the namelist, we follow [38] and set

$$b(h) = \begin{cases} \frac{2}{G^*}(1 - \frac{G(h)}{G^*}) & \text{if } G(h) < G^* \\ 0 & \text{otherwise} \end{cases} \tag{38}$$

where $G(h)$ is the fractional area covered by ice thinner than h, and G^* is an empirical constant. Integrating $a_P(h)$ between category boundaries H_{n-1} and H_n, we obtain the mean value of a_P in category n:

$$a_{Pn} = \frac{2}{G^*}(G_n - G_{n-1})\left(1 - \frac{G_{n-1} + G_n}{2G^*}\right), \tag{39}$$

where a_{Pn} is the ratio of the ice area ridging (or open water area closing) in category n to the total area ridging and closing, and G_n is the total fractional ice area in categories 0 to n. Equation (39) applies to categories with $G_n < G^*$. If $G_{n-1} < G^* < G_n$, then (39) is valid with G^* replacing G_n, and if $G_{n-1} > G^*$, then $a_{Pn} = 0$. If the open water fraction $a_0 > G^*$, no ice can ridge, because "ridging" simply reduces the area of open water. As in [38] we set $G^* = 0.15$.

If the spatial resolution is too fine for a given time step Δt, the weighting function (38) can promote numerical instability. For $\Delta t = 1$ hour, resolutions finer than $\Delta x \sim 10$ km are typically unstable. The instability results from feedback between the ridging scheme and the dynamics via the ice strength. If the strength changes significantly on time scales less than Δt, the EVP solution is inaccurate and sometimes oscillatory. As a result, the fields of ice area, thickness, velocity, strength, divergence, and shear can become noisy and unphysical.

A more stable weighting function was suggested by [23]:

$$b(h) = \frac{\exp[-G(h)/a^*]}{a^*[1 - \exp(-1/a^*)]} \tag{40}$$

When integrated between category boundaries, (40) implies

$$a_{Pn} = \frac{\exp(-G_n/a^*) - \exp(-G_{n-1}/a^*)}{1 - \exp(-1/a^*)} \tag{41}$$

This weighting function is used if $\texttt{krdg}_{\texttt{partic}} = \texttt{1}$ in the namelist. From (40), the mean value of G for ice participating in ridging is a^*, as compared to $G^*/3$ for (38). For typical ice thickness distributions, setting $a^* = 0.05$ with $\texttt{krdg}_{\texttt{partic}} = \texttt{1}$ gives participation fractions similar to those given by $G^* = 0.15$ with $\texttt{krdg}_{\texttt{partic}} = \texttt{0}$. See [23] for a detailed comparison of these two participation functions.

Thin ice is converted to thick ridged ice in a way that reduces the total ice area while conserving ice volume and internal energy. There are two namelist options for redistributing ice among thickness categories. If $\texttt{krdg}_{\texttt{redist}} = \texttt{0}$, ridging ice of thickness h_n forms ridges whose area is distributed uniformly between $H_{\min} = 2h_n$ and $H_{\max} = 2\sqrt{H^* h_n}$, as in [12]. The default value of H^* is 25 m, as in earlier versions of CICE. Observations suggest that $H^* = 50$ m gives a better fit to first-year ridges [1], although the lower

value may be appropriate for multiyear ridges [9]. The ratio of the mean ridge thickness to the thickness of ridging ice is $k_n = (H_{\min} + H_{\max})/(2h_n)$. If the area of category n is reduced by ridging at the rate r_n, the area of thicker categories grows simultaneously at the rate r_n/k_n. Thus the *net* rate of area loss due to ridging of ice in category n is $r_n(1 - 1/k_n)$.

The ridged ice area and volume are apportioned among categories in the thickness range $(H_{\min}, H_{\max})$. The fraction of the new ridge area in category m is

$$f_m^{\text{area}} = \frac{H_R - H_L}{H_{\max} - H_{\min}}, \tag{42}$$

where $H_L = \max(H_{m-1}, H_{\min})$ and $H_R = \min(H_m, H_{\max})$. The fraction of the ridge volume going to category m is

$$f_m^{\text{vol}} = \frac{(H_R)^2 - (H_L)^2}{(H_{\max})^2 - (H_{\min})^2}. \tag{43}$$

This uniform redistribution function tends to produce too little ice in the 3–5 m range and too much ice thicker than 10 m [1]. Observations show that the ITD of ridges is better approximated by a negative exponential. Setting $\texttt{krdg}_{\texttt{redist}} = \texttt{1}$ gives ridges with an exponential ITD [23]:

$$g_R(h) \propto \exp[-(h - H_{\min})/\lambda] \tag{44}$$

for $h >= H_{\min}$, with $g_R(h) = 0$ for $h < H_{\min}$. Here, λ is an empirical *e*-folding scale and $H_{\min} = 2h_n$ (where h_n is the thickness of ridging ice). We assume that $\lambda = \mu h_n^{1/2}$, where μ is a tunable parameter with units $\text{m}^{1/2}$. Thus the mean ridge thickness increases in proportion to $h_n^{1/2}$, as in [12]. The default value $\mu = 4.0\ \text{m}^{1/2}$ gives λ in the range 1–4 m for most ridged ice. Ice strengths with $\mu = 4.0\ \text{m}^{1/2}$ and $\texttt{krdg}_{\texttt{redist}} = \texttt{1}$ are roughly comparable to the strengths with $H^* = 50$ and $\texttt{krdg}_{\texttt{redist}} = \texttt{0}$.

From (44) it can be shown that the fractional area going to category m as a result of ridging is

$$f_m^{\text{area}} = \exp[-(H_{m-1} - H_{\min})/\lambda] - \exp[-(H_m - H_{\min})/\lambda]. \tag{45}$$

The fractional volume going to category m is

$$f_m^{\text{vol}} = \frac{(H_{m-1} + \lambda)\exp[-(H_{m-1} - H_{\min})/\lambda] - (H_m + \lambda)\exp[-(H_m - H_{\min})/\lambda]}{H_{min} + \lambda}. \tag{46}$$

Equations (45) and (46) replace (42) and (43) when $\texttt{krdg}_{\texttt{redist}} = \texttt{1}$.

Internal ice energy is transferred between categories in proportion to ice volume. Snow volume and internal energy are transferred in the same way, except that a fraction of the snow may be deposited in the ocean instead of added to the new ridge.

The net area removed by ridging and closing is a function of the strain rates. Let R_{net} be the net rate of area loss for the ice pack (i.e., the rate of open water area closing, plus the net rate of ice area loss due to ridging). Following [9], R_{net} is given by

$$R_{\text{net}} = \frac{C_s}{2}(\Delta - |D_D|) - \min(D_D, 0), \tag{47}$$

where C_s is the fraction of shear dissipation energy that contributes to ridge-building, D_D is the divergence, and Δ is a function of the divergence and shear. These strain rates are computed by the dynamics scheme. The default value of C_s is 0.25.

Next, define $R_{\text{tot}} = \sum_{n=0}^{N} r_n$. This rate is related to R_{net} by

$$R_{\text{net}} = \left[a_{P0} + \sum_{n=1}^{N} a_{Pn}\left(1 - \frac{1}{k_n}\right)\right] R_{\text{tot}}. \tag{48}$$

Given $R_{\rm net}$ from (47), we use (48) to compute $R_{\rm tot}$. Then the area ridged in category n is given by $a_{rn} = r_n \Delta t$, where $r_n = a_{Pn} R_{\rm tot}$. The area of new ridges is a_{rn}/k_n, and the volume of new ridges is $a_{rn} h_n$ (since volume is conserved during ridging). We remove the ridging ice from category n and use (42) and (43) (or 45) and (46)) to redistribute the ice among thicker categories.

Occasionally the ridging rate in thickness category n may be large enough to ridge the entire area a_n during a time interval less than Δt. In this case $R_{\rm tot}$ is reduced to the value that exactly ridges an area a_n during Δt. After each ridging iteration, the total fractional ice area a_i is computed. If $a_i > 1$, the ridging is repeated with a value of $R_{\rm net}$ sufficient to yield $a_i = 1$.

The ice strength P may be computed in either of two ways. If the namelist parameter $\texttt{krdg}_{\texttt{redist}} = \texttt{0}$, we use the strength formula from [11]:

$$P = P^* h \exp[-C(1 - a_i)], \tag{49}$$

where $P^* = 27,500\,\mathrm{N/m}$ and $C = 20$ are empirical constants, and h is the mean ice thickness. Alternatively, setting $\texttt{krdg}_{\texttt{redist}} = \texttt{1}$ gives an ice strength closely related to the ridging scheme. Following [30], the strength is assumed proportional to the change in ice potential energy ΔE_P per unit area of compressive deformation. Given uniform ridge ITDs ($\texttt{krdg}_{\texttt{redist}} = \texttt{0}$), we have

$$P = C_f\, C_p\, \beta \sum_{n=1}^{N_C} \left[-a_{Pn}\, h_n^2 + \frac{a_{Pn}}{k_n} \left(\frac{(H_n^{\max})^3 - (H_n^{\min})^3}{3(H_n^{\max} - H_n^{\min})} \right) \right], \tag{50}$$

where $C_P = (g/2)(\rho_i/\rho_w)(\rho_w - \rho_i)$, $\beta = R_{\rm tot}/R_{\rm net} > 1$ from (48), and C_f is an empirical parameter that accounts for frictional energy dissipation. Following [9], we set $C_f = 17$. The first term in the summation is the potential energy of ridging ice, and the second, larger term is the potential energy of the resulting ridges. The factor of β is included because a_{Pn} is normalized with respect to the total area of ice ridging, not the net area removed. Recall that more than one unit area of ice must be ridged to reduce the net ice area by one unit. For exponential ridge ITDs ($\texttt{krdg}_{\texttt{redist}} = \texttt{1}$), the ridge potential energy is modified:

$$P = C_f\, C_p\, \beta \sum_{n=1}^{N_C} \left[-a_{Pn}\, h_n^2 + \frac{a_{Pn}}{k_n} \left(H_{\min}^2 + 2H_{\min}\lambda + 2\lambda^2 \right) \right] \tag{51}$$

The energy-based ice strength given by (50) or (51) is more physically realistic than the strength given by (49). However, use of (49) is less likely to allow numerical instability at a given resolution and time step. See [23] for more details.

3.4 Dynamics

The elastic-viscous-plastic (EVP) model represents a modification of the standard viscous-plastic (VP) model for sea ice dynamics [11]. It reduces to the VP model at time scales associated with the wind forcing, while at shorter time scales the adjustment process takes place by a numerically more efficient elastic wave mechanism. While retaining the essential physics, this elastic wave modification leads to a fully explicit numerical scheme which greatly improves the model's computational efficiency.

The EVP sea ice dynamics model is thoroughly documented in [14], [13], [15] and [16]. Simulation results and performance of this model have been compared with the VP model in realistic simulations of the Arctic [17]. Here we summarize the equations and direct the reader to the above references for details. The numerical implementation in this code release is that of [15] and [16].

The force balance per unit area in the ice pack is given by a two-dimensional momentum equation [11], obtained by integrating the 3D equation through the thickness of the ice in the vertical direction:

$$m \frac{\partial \mathbf{u}}{\partial t} = \nabla \cdot \sigma + \vec{\tau}_a + \vec{\tau}_w - \hat{k} \times m f \mathbf{u} - mg \nabla H_\circ, \tag{52}$$

where m is the combined mass of ice and snow per unit area and $\vec{\tau}_a$ and $\vec{\tau}_w$ are wind and ocean stresses, respectively. The strength of the ice is represented by the internal stress tensor σ_{ij}, and the other two terms on the right hand side are stresses due to Coriolis effects and the sea surface slope. The parameterization for the wind and ice-ocean stress terms must contain the ice concentration as a multiplicative factor to be consistent with the formal theory of free drift in low ice concentration regions. A careful explanation of the issue and its continuum solution is provided in [16] and [4].

For convenience we formulate the stress tensor σ in terms of $\sigma_1 = \sigma_{11} + \sigma_{22}$, $\sigma_2 = \sigma_{11} - \sigma_{22}$, and introduce the divergence, D_D, and the horizontal tension and shearing strain rates, D_T and D_S respectively. The internal stress tensor is determined from a regularized version of the VP constitutive law,

$$\frac{1}{E}\frac{\partial \sigma_1}{\partial t} + \frac{\sigma_1}{2\zeta} + \frac{P}{2\zeta} = D_D, \tag{53}$$

$$\frac{1}{E}\frac{\partial \sigma_2}{\partial t} + \frac{\sigma_2}{2\eta} = D_T, \tag{54}$$

$$\frac{1}{E}\frac{\partial \sigma_{12}}{\partial t} + \frac{\sigma_{12}}{2\eta} = \frac{1}{2}D_S, \tag{55}$$

where

$$D_D = \dot{\epsilon}_{11} + \dot{\epsilon}_{22}, \tag{56}$$

$$D_T = \dot{\epsilon}_{11} - \dot{\epsilon}_{22}, \tag{57}$$

$$D_S = 2\dot{\epsilon}_{12}, \tag{58}$$

$$\dot{\epsilon}_{ij} = \frac{1}{2}\left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i}\right),$$

$$\zeta = \frac{P}{2\Delta},$$

$$\eta = \frac{P}{2\Delta e^2},$$

$$\Delta = \left[D_D^2 + \frac{1}{e^2}\left(D_T^2 + D_S^2\right)\right]^{1/2},$$

and P is a function of the ice thickness and concentration, described in Section 3.3. The dynamics component employs a "replacement pressure" (see [10], for example), which serves to prevent residual ice motion due to spatial variations of P when the rates of strain are exactly zero.

Several changes have been made to the EVP model since the original release. In the earlier version, the viscosities were held fixed while the stress and momentum equations were subcycled with the smaller time step `dte`. The reason for implementing the EVP model in this way was to reproduce the results of the original VP model as closely as possible. When solved with time steps of several hours or more, the VP model suffers a linearization error associated with the viscosities, which are lagged over the time step [13]. This led to principal stress states that were widely scattered outside the elliptical yield curve in both models [17]. We have addressed this problem by updating the viscosities during the subcycling, so that the entire dynamics component is subcycled within the time step. Taken alone, this change would require an increased number operations to compute the viscosities.

However, the new dynamics component is roughly as efficient as the earlier version due to a change in the definition of the elastic parameter E. E is now defined in terms of a damping timescale T for elastic waves, $\Delta t_e < T < \Delta t$, as

$$E = \frac{\zeta}{T},$$

where $T = E_\circ \Delta t$ and $E_\circ$ (`eyc`) is a tunable parameter less than one, as before. The stress equations (53–55) become

$$\begin{aligned} \frac{\partial \sigma_1}{\partial t} + \frac{\sigma_1}{2T} + \frac{P}{2T} &= \frac{P}{2T\Delta} D_D, \\ \frac{\partial \sigma_2}{\partial t} + \frac{e^2 \sigma_2}{2T} &= \frac{P}{2T\Delta} D_T, \\ \frac{\partial \sigma_{12}}{\partial t} + \frac{e^2 \sigma_{12}}{2T} &= \frac{P}{4T\Delta} D_S. \end{aligned}$$

All coefficients on the left-hand side are constant except for P, which changes only on the longer time step Δt. This modification compensates for the decreased efficiency of including the viscosity terms in the subcycling. (Note that the viscosities do not appear explicitly.) Choices of the parameters used to define E, T and Δt_e are discussed in Section 4.4.

A different discretization of the stress terms $\partial \sigma_{ij}/\partial x_j$ in the momentum equation is now used, which enabled the discrete equations to be derived from the continuous equations written in curvilinear coordinates. In this manner, metric terms associated with the curvature of the grid were incorporated into the discretization explicitly. We no longer use a "triangle discretization," in which the strain rates and stresses were constant over each of four subtriangles within each grid cell, and instead use a bilinear approximation for the velocities and stresses. Details pertaining to the spatial discretization are found in [15].

The momentum equation is discretized in time as follows. First, for clarity, the two components of (52) are

$$\begin{aligned} m\frac{\partial u}{\partial t} &= \frac{\partial \sigma_{1j}}{x_j} + \tau_{ax} + a_i c_w \rho_w \left|\mathbf{U}_w - \mathbf{u}\right| \left[(U_w - u)\cos\theta - (V_w - v)\sin\theta\right] + mfv - mg\frac{\partial H_\circ}{\partial x}, \\ m\frac{\partial v}{\partial t} &= \frac{\partial \sigma_{2j}}{x_j} + \tau_{ay} + a_i c_w \rho_w \left|\mathbf{U}_w - \mathbf{u}\right| \left[(U_w - u)\sin\theta - (V_w - v)\cos\theta\right] - mfu - mg\frac{\partial H_\circ}{\partial y}. \end{aligned}$$

In the code, $\texttt{vrel} = a_i c_w \rho_w \left|\mathbf{U}_w - \mathbf{u}^k\right|$, where k denotes the subcycling step. The following equations illustrate the time discretization and define some of the other variables used in the code.

$$\underbrace{\left(\frac{m}{\Delta t} + \texttt{vrel}\cos\theta\right)}_{\texttt{cca}} u^{k+1} - \underbrace{(mf + \texttt{vrel}\sin\theta)}_{\texttt{ccb}} v^{k+1} = \underbrace{\frac{\partial \sigma_{1j}^{k+1}}{x_j}}_{\texttt{strintx}} + \underbrace{\tau_{ax} - mg\frac{\partial H_\circ}{\partial x}}_{\texttt{forcex}} + \texttt{vrel}\underbrace{(U_w\cos\theta - V_w\sin\theta)}_{\texttt{waterx}} + \frac{m}{\Delta t}u^k,$$

$$\underbrace{(mf + \texttt{vrel}\sin\theta)}_{\texttt{ccb}} u^{k+1} + \underbrace{\left(\frac{m}{\Delta t} + \texttt{vrel}\cos\theta\right)}_{\texttt{cca}} v^{k+1} = \underbrace{\frac{\partial \sigma_{2j}^{k+1}}{x_j}}_{\texttt{strinty}} + \underbrace{\tau_{ay} - mg\frac{\partial H_\circ}{\partial x}}_{\texttt{forcey}} + \texttt{vrel}\underbrace{(U_w\sin\theta + V_w\cos\theta)}_{\texttt{watery}} + \frac{m}{\Delta t}v^k,$$

and `vrel·waterx(y) = taux(y)`. We solve this system of equations analytically for u^{k+1} and v^{k+1}. When the subcycling is finished for each (thermodynamic) time step, the ice-ocean stress must be constructed from `taux(y)` and the terms containing `vrel` on the left hand side of the equations. This is done in subroutine *evp_finish*.

3.5 Thermodynamics

The thermodynamic sea ice model is based on [27] and [3], and is described more fully in [20]. For each thickness category the model computes changes in the ice and snow thickness and vertical temperature profile resulting from radiative, turbulent, and conductive heat fluxes. The ice has a temperature-dependent specific heat to simulate the effect of brine pocket melting and freezing.

Each thickness category n in each grid cell is treated as a horizontally uniform column with ice thickness $h_{in} = v_{in}/a_{in}$ and snow thickness $h_{sn} = v_{sn}/a_{in}$. (Henceforth we omit the category index n.) Each column is divided into N_i ice layers of thickness $\Delta h_i = h_i/N_i$ and, if snow is present, a single snow layer. (We will allow for multiple snow layers in future versions of CICE.) The surface temperature (i.e., the temperature of ice or snow at the interface with the atmosphere) is T_{sf}, which cannot exceed 0°C. The temperature at the midpoint of the snow layer is T_s, and the midpoint ice layer temperatures are T_{ik}, where k ranges from 1 to N_i. The temperature at the bottom of the ice is held at T_f, the freezing temperature of the ocean mixed layer. All temperatures are in degrees Celsius unless otherwise specified.

The vertical salinity profile is prescribed and is unchanging in time. The snow is assumed to be fresh, and the midpoint salinity S_{ik} in each ice layer is given by

$$S_{ik} = \frac{1}{2} S_{\max}[1 - \cos(\pi z^{(\frac{a}{z+b})})], \tag{59}$$

where $z \equiv (k - 1/2)/N_i$, $S_{\max} = 3.2$ psu, and $a = 0.407$ and $b = 0.573$ are determined from a least-squares fit to the salinity profile observed in multiyear sea ice by [31]. This profile varies from $S = 0$ at the top surface ($z = 0$) to $S = S_{\max}$ at the bottom surface ($z = 1$) and is similar to that used by [27]. Equation (59) is fairly accurate for ice that has drained at the top surface due to summer melting. It is not a good approximation for cold first-year ice, which has a more vertically uniform salinity because it has not yet drained. However, the effects of salinity on heat capacity are small for temperatures well below freezing, so the salinity error does not lead to significant temperature errors.

Each ice layer has an enthalpy q_{ik}, defined as the negative of the energy required to melt a unit volume of ice and raise its temperature to 0°C. Because of internal melting and freezing in brine pockets, the ice enthalpy depends on the brine pocket volume and is a function of temperature and salinity. Since the salinity is prescribed, there is a one-to-one relationship between temperature and enthalpy. We can also define a snow enthalpy q_s, which depends on temperature alone. Expressions for the enthalpy are derived in Section 3.5.3.

Given surface forcing at the atmosphere-ice and ice-ocean interfaces along with the ice and snow thicknesses and temperatures/enthalpies at time m, the thermodynamic model advances these quantities to time $m + 1$. The calculation proceeds in two steps. First we solve a set of equations for the new temperatures, as discussed in Section 3.5.2. Then we compute the melting, if any, of ice or snow at the top surface, and the growth or melting of ice at the bottom surface, as described in Section 3.5.3. We begin by describing the surface forcing parameterizations, which are closely related to the ice and snow surface temperatures.

3.5.1 Thermodynamic surface forcing

The net energy flux from the atmosphere to the ice (with all fluxes defined as positive downward) is

$$F_0 = F_s + F_l + F_{L\downarrow} + F_{L\uparrow} + (1 - \alpha)(1 - i_0)F_{sw},$$

where F_s is the sensible heat flux, F_l is the latent heat flux, $F_{L\downarrow}$ is the incoming longwave flux, $F_{L\uparrow}$ is the outgoing longwave flux, F_{sw} is the incoming shortwave flux, α is the shortwave albedo, and i_0 is the fraction of absorbed shortwave flux that penetrates into the ice.

The albedo depends on the temperature and thickness of ice and snow and on the spectral distribution of the incoming solar radiation. Albedo parameters have been chosen to fit observations from the SHEBA field experiment. For $T_{sf} < -1^\circ$C and $h_i > 0.5$ m, the bare ice albedo is 0.78 for visible wavelengths (< 700 nm) and 0.36 for near IR wavelengths (> 700 nm). As h_i decreases from 0.5 m to zero, the ice albedo decreases smoothly (using an arctangent function) to the ocean albedo, 0.06. The ice albedo in both spectral bands decreases by 0.075 as T_{sf} rises from -1°C to 0°C. The albedo of cold snow ($T_{sf} < -1^\circ$C) is 0.98 for visible wavelengths and 0.70 for near IR wavelengths. The visible snow albedo decreases by 0.10

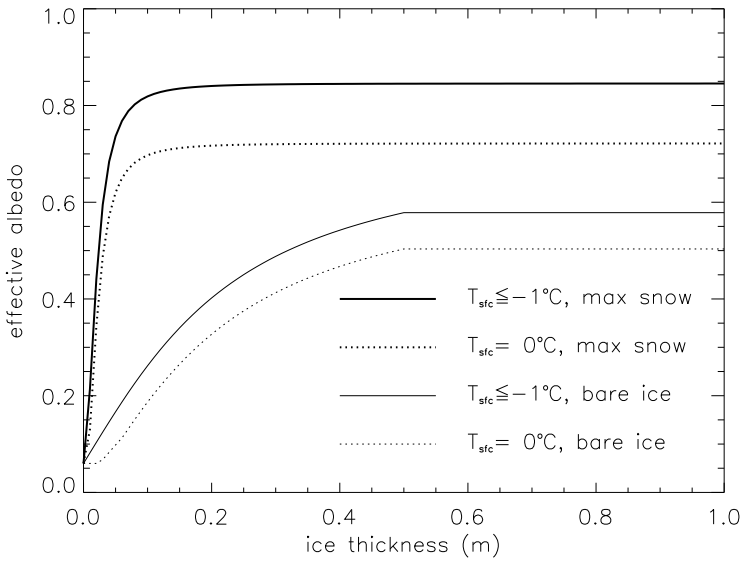

Figure 5: Albedo as a function of ice thickness and temperature, for the two extrema in snow depth. Maximum snow depth is computed based on Archimedes' Principle for the given ice thickness. These curves represent the envelope of possible albedo values.

and the near IR albedo by 0.15 as T_{sf} increases from $-1°\mathrm{C}$ to $0°\mathrm{C}$. The total albedo is an area-weighted average of the ice and snow albedos, where the fractional snow-covered area is

$$f_{snow} = \frac{h_s}{h_s + h_{snowpatch}},$$

and $h_{snowpatch} = 0.02$ m. The envelope of albedo values is shown in Figure 5.

The net absorbed shortwave flux is $F_{swabs} = \sum(1 - \alpha_j)F_{sw\downarrow j}$, where the summation is over four radiative categories (direct and diffuse visible, direct and diffuse near infrared). The flux penetrating into the ice is $I_0 = i_0\, F_{swabs}$, where $i_0 = 0.70\,(1 - f_{snow})$ for visible radiation and $i_0 = 0$ for near IR. Radiation penetrating into the ice is attenuated according to Beer's Law:

$$I(z) = I_0 \exp(-\kappa_i z), \tag{60}$$

where $I(z)$ is the shortwave flux that reaches depth z beneath the surface without being absorbed, and κ_i is the bulk extinction coefficient for solar radiation in ice, set to $1.4\ \mathrm{m}^{-1}$ for visible wavelengths [8]. A fraction $\exp(-\kappa_i h_i)$ of the penetrating solar radiation passes through the ice to the ocean ($F_{sw\Downarrow}$).

While incoming shortwave and longwave radiation are obtained from the atmosphere, outgoing longwave radiation and the turbulent heat fluxes are derived quantities. Outgoing longwave takes the standard blackbody form, $F_{L\uparrow} = \epsilon\sigma\left(T_{sf}^K\right)^4$, where $\epsilon = 0.95$ is the emissivity of snow or ice, σ is the Stefan-Boltzmann constant and T_{sf}^K is the surface temperature in Kelvin. (The longwave fluxes are partitioned such that $\epsilon F_{L\downarrow}$ is absorbed at the surface, the remaining $(1 - \epsilon)\, F_{L\downarrow}$ being returned to the atmosphere via $F_{L\uparrow}$.) The sensible heat flux is proportional to the difference between air potential temperature and the surface temperature of the snow or snow-free ice,

$$F_s = C_s\left(\Theta_a - T_{sf}^K\right).$$

C_s and C_l (below) are nonlinear turbulent heat transfer coefficients described in Section 2.1. Similarly, the latent heat flux is proportional to the difference between Q_a and the surface saturation specific humidity Q_{sf}:

$$\begin{aligned} F_l &= C_l\,(Q_a - Q_{sf})\,, \\ Q_{sf} &= (q_1/\rho_a)\exp(-q_2/T_{sf}^K), \end{aligned}$$

where $q_1 = 1.16378 \times 10^7\,\mathrm{kg/m^3}$, $q_2 = 5897.8\,\mathrm{K}$, T_{sf}^K is the surface temperature in Kelvin, and ρ_a is the surface air density.

The net downward heat flux from the ice to the ocean is given by [25]:

$$F_{bot} = -\rho_w c_w c_h u_* (T_w - T_f), \tag{61}$$

where ρ_w is the density of seawater, c_w is the specific heat of seawater, $c_h = 0.006$ is a heat transfer coefficient, $u_* = \sqrt{|\vec{\tau}_w|\,/\rho_w}$ is the friction velocity, and T_w is the sea surface temperature. The minimum value of u_* depends on whether the model is run coupled; lack of currents in uncoupled runs mean that not enough heat is available to melt ice in the standard formulation. In this release we have $u_{*\,\min} = 5 \times 10^{-3}$ for coupled runs and 5×10^{-2} for uncoupled runs.

F_{bot} is limited by the total amount of heat available from the ocean, F_{frzmlt}. Additional heat, F_{side}, is used to melt the ice laterally following [26] and [35]. F_{bot} and the fraction of ice melting laterally are scaled so that $F_{bot} + F_{side} \geq F_{frzmlt}$ in the case that $F_{frzmlt} < 0$ (melting).

3.5.2 New temperatures

Given the temperatures T_{sf}^m, T_s^m, and T_{ik}^m at time m, we solve a set of finite-difference equations to obtain the new temperatures at time $m+1$. Each temperature is coupled to the temperatures of the layers immediately above and below by heat conduction terms that are treated implicitly. For example, the rate of change of T_{ik} depends on the new temperatures in layers $k-1$, k, and $k+1$. Thus we have a set of equations of the form

$$\mathbf{A}\mathbf{x} = \mathbf{b}, \tag{62}$$

where $\mathbf{A}$ is a tridiagonal matrix, $\mathbf{x}$ is a column vector whose components are the unknown new temperatures, and $\mathbf{b}$ is another column vector. Given $\mathbf{A}$ and $\mathbf{b}$, we can compute $\mathbf{x}$ with a standard tridiagonal solver.

There are four general cases: (1) $T_{sf} < 0°\mathrm{C}$, snow present; (2) $T_{sf} = 0°\mathrm{C}$, snow present; (3) $T_{sf} < 0°\mathrm{C}$, snow absent; and (4) $T_{sf} = 0°\mathrm{C}$, snow absent. For case 1 we have one equation (the top row of the matrix) for the new surface temperature, one equation (the second row) for the new snow temperature, and N_i equations (the remaining rows) for the new ice temperatures. For cases 2 and 4 we omit the equation for the surface temperature, which is held at $0°\mathrm{C}$, and for cases 3 and 4 we omit the snow temperature equation.

The rate of temperature change in the ice interior is given by [27]:

$$\rho_i c_i \frac{\partial T_i}{\partial t} = \frac{\partial}{\partial z}\left(k_i \frac{\partial T_i}{\partial z}\right) - \frac{\partial}{\partial z}[I_0 \exp(-\kappa_i z)], \tag{63}$$

where $\rho_i = 917\ \mathrm{kg/m^3}$ is the sea ice density (assumed to be uniform), $c_i(T,S)$ is the specific heat of sea ice, $k_i(T,S)$ is the thermal conductivity of sea ice, I_0 is the flux of penetrating solar radiation, attenuated with extinction coefficient κ_i (see previous section), and z is the vertical coordinate, defined to be positive downward with $z = 0$ at the top surface. The specific heat of sea ice is given to an excellent approximation by [29]

$$c_i(T,S) = c_0 + \frac{L_0 \mu S}{T^2}, \tag{64}$$

where $c_0 = 2106$ J/kg/deg is the specific heat of fresh ice at 0°C, $L_0 = 3.34 \times 10^5$ J/kg is the latent heat of fusion of fresh ice at 0°C, and $\mu = 0.054$ deg/psu is the ratio between the freezing temperature and salinity of brine. Following [39], the thermal conductivity is given by

$$k_i(T, S) = k_0 + \frac{\beta S}{T}, \tag{65}$$

where $k_0 = 2.03$ W/m/deg is the conductivity of fresh ice and $\beta = 0.13$ W/m/psu is an empirical constant.

The corresponding equation for the temperature change in snow is

$$\rho_s c_s \frac{\partial T_s}{\partial t} = \frac{\partial}{\partial z}\left(k_s \frac{\partial T_s}{\partial z}\right), \tag{66}$$

where $\rho_s = 330$ kg/m^3 is the snow density (also assumed uniform), $c_s = c_0$ is the specific heat of snow, and $k_s = 0.30$ W/m/deg is the thermal conductivity of snow. Penetrating solar radiation is neglected in (66) because most of the incoming sunlight is absorbed near the top surface when snow is present.

We now convert (63) and (66) to finite-difference form. The resulting equations are second-order accurate in space, except possibly at material boundaries, and first-order accurate in time. Before writing the equations in full we give finite-difference expressions for some of the terms.

First consider the terms on the left-hand side of (63) and (66). We write the time derivatives as

$$\frac{\partial T}{\partial t} = \frac{T^{m+1} - T^m}{\Delta t},$$

where T is the temperature of either ice or snow. The specific heat of ice layer k is approximated as

$$c_{ik} = c_0 + \frac{L_0 \mu S_{ik}}{T_{ik}^m \, T_{ik}^{m+1}}, \tag{67}$$

which ensures that energy is conserved during a change in temperature. This can be shown by using (64) to integrate $c_i \, dT$ from T_{ik}^m to T_{ik}^{m+1}; the result is $c_{ik}(T_{ik}^{m+1} - T_{ik}^m)$, where c_{ik} is given by (67). Unfortunately, the specific heat is a nonlinear function of T_{ik}^{m+1}, the unknown new temperature. We can retain a set of linear equations, however, by initially guessing $T_{ik}^{m+1} = T_{ik}^m$ and then iterating the solution, updating T_{ik}^{m+1} in (67) with each iteration until the solution converges.

Next consider the first term on the right-hand side of (63): the heat diffusion term. In the ice interior (layers 2 to $N_i - 1$) this term is discretized as

$$\frac{\partial}{\partial z}\left(k_i \frac{\partial T_i}{\partial z}\right) = \frac{1}{\Delta h_i}\left[\frac{k_{i,k}(T_{i,k-1}^{m+1} - T_{ik}^{m+1})}{\Delta h_i} - \frac{k_{i,k+1}(T_{ik}^{m+1} - T_{i,k+1}^{m+1})}{\Delta h_i}\right], \tag{68}$$

where k_{ik} is the thermal conductivity at the upper boundary of layer k. The approximation in (68) is spatially centered and second-order accurate. We can write similar expressions for heat diffusion in the top and bottom ice layers and the snow layer, as shown below. Note that the conduction terms are treated implicitly; that is, they depend on the temperatures at the new time $m+1$. The resulting scheme is first-order accurate in time and unconditionally stable.

Using (65), we approximate k_{ik} in the ice interior (at the upper boundary of layers 2 to N_i) as

$$k_{ik} = k_0 + \frac{\beta\,(S_{i,k-1} + S_{ik})}{T_{i,k-1}^m + T_{ik}^m}.$$

Since the conductivity does not depend as strongly on temperature as does the specific heat, we define k_{ik} in terms of the ice temperatures at time m. Thus the conductivity does not have to be updated with each iteration. At the bottom surface we have

$$k_{i,N_i+1} = k_0 + \frac{\beta S_{\max}}{T_f}.$$

The conductivity at the top ice surface, k_{i1}, depends on whether snow is present. If there is no snow, we set

$$k_{i1} = k_0 + \frac{\beta S_{i1}}{T_{i1}^m}.$$

(We avoid defining k_{i1} in terms of T_{sf} because then it would be undefined for $T_{sf} = 0$.) If snow is present we assume a continuous heat flux across the ice-snow interface:

$$k_{i1}\frac{T_{i1}^m - T_{int}^m}{\Delta h_i/2} = k_s\frac{T_{int}^m - T_s^m}{h_s/2},$$

where T_{int} is the interface temperature. Solving for T_{int}^m, we can show that this heat flux is equivalent to

$$k_{int}\,\frac{T_{i1}^m - T_s^m}{(\Delta h_i + h_s)/2},$$

where k_{int}, the equivalent conductivity at the interface, is defined as

$$k_{int} = \frac{k_{i1}k_s(\Delta h_i + h_s)}{h_s k_{i1} + \Delta h_i k_s}.$$

Finally, consider the second term on the right in (63). From (60), the fraction of the penetrating solar radiation I_0 transmitted through layer k without being absorbed is

$$\tau_k = \exp(-\kappa_i k \Delta h_i).$$

Thus the flux absorbed in layer k is

$$Q_k = I_0(\tau_{k-1} - \tau_k).$$

The flux absorbed per unit ice thickness is $Q_k/\Delta h_i$, the desired finite-difference approximation to

$$-\frac{\partial}{\partial z}[I_0 \exp(-\kappa_i z)].$$

We now construct a system of equations for the new temperatures. (The reader uninterested in algebraic details may prefer to skip to the next section.) We begin at the surface and work down. For case 1 ($T_{sf} < 0°\mathrm{C}$ and snow present), we require

$$F_0 = F_{ct}, \tag{69}$$

where F_{ct} is the conductive flux from the top surface to the ice interior, and both fluxes are evaluated at time $m + 1$. Although F_0 is a nonlinear function of T_{sf}, we can make the linear approximation

$$F_0^{m+1} = F_0^* + \left(\frac{dF_0}{dT_{sf}}\right)^* (T_{sf}^{m+1} - T_{sf}^*),$$

where T_{sf}^* is the surface temperature from the most recent iteration, and F_0^* and $(dF_0/dT_{sf})^*$ are functions of T_{sf}^*. We initialize $T_{sf}^* = T_{sf}^m$ and update it with each iteration. Thus we can rewrite (69) as

$$F_0^* + \left(\frac{dF_0}{dT_{sf}}\right)^* (T_{sf}^{m+1} - T_{sf}^*) = K_s(T_{sf}^{m+1} - T_s^{m+1}),$$

where $K_s \equiv 2k_s/h_s$. Rearranging terms, we obtain

$$\left[\left(\frac{dF_0}{dT_{sf}}\right)^* - K_s\right] T_{sf}^{m+1} + K_s T_s^{m+1} = \left(\frac{dF_0}{dT_{sf}}\right)^* T_{sf}^* - F_0^*, \tag{70}$$

the first equation in the set of equations (62).

Continuing with case 1, we write the equation for the change in T_s:

$$\rho_s c_s \frac{(T_s^{m+1} - T_s^m)}{\Delta t} = \frac{1}{h_s}[K_s(T_{sf}^{m+1} - T_s^{m+1}) - K_{int}(T_s^{m+1} - T_{i1}^{m+1})] \tag{71}$$

where $K_{int} \equiv 2k_{int}/(\Delta h_i + h_s)$. In tridiagonal matrix form (71) becomes

$$-\eta_s K_s T_{sf}^{m+1} + [1 + \eta_s(K_{int} + K_s)]T_s^{m+1} - \eta_s K_{int} T_{i1}^{m+1} = T_s^m, \tag{72}$$

where $\eta_s \equiv \Delta t/(\rho_s c_s h_s)$.

The ice equations for the top layer, the interior layers (2 to $N_i - 1$), and the bottom layer, respectively, are

$$\begin{aligned}
\rho_i c_{i1} \frac{(T_{i1}^{m+1} - T_{i1}^m)}{\Delta t} &= \frac{1}{\Delta h_i}[K_{int}(T_s^{m+1} - T_{i1}^{m+1}) - K_{i2}(T_{i1}^{m+1} - T_{i2}^{m+1}) + Q_1], \\
\rho_i c_{ik} \frac{(T_{ik}^{m+1} - T_{ik}^m)}{\Delta t} &= \frac{1}{\Delta h_i}[K_{ik}(T_{i,k-1}^{m+1} - T_{ik}^{m+1}) - K_{i,k+1}(T_{ik}^{m+1} - T_{i,k+1}^{m+1}) + Q_k], \\
\rho_i c_{iN} \frac{(T_{iN}^{m+1} - T_{iN}^m)}{\Delta t} &= \frac{1}{\Delta h_i}\{K_{iN}(T_{i,N-1}^{m+1} - T_{iN}^{m+1}) - \\
&\qquad K_{i,N+1}[\gamma_1(T_{iN}^{m+1} - T_f) + \gamma_2(T_{i,N-1}^{m+1} - T_f)] + Q_1\},
\end{aligned}$$

where $K_{ik} \equiv k_{ik}/\Delta h_i$ and $N \equiv N_i$. The coefficients $\gamma_1 = 3$ and $\gamma_2 = -1/3$ provide one-sided second-order spatial accuracy at the bottom surface; they are found from a Taylor series expansion of dT/dz at $z = h_i$. Rearranging terms, we obtain

$$-\eta_{i1} K_{int} T_s^{m+1} + [1 + \eta_{i1}(K_{int} + K_{i2})]T_{i1}^{m+1} - \eta_{i1} K_{i2} T_{i2}^{m+1} = T_{i1}^m + \eta_{i1} Q_1, \tag{73}$$

$$-\eta_{ik} K_{ik} T_{i,k-1}^{m+1} + [1 + \eta_{ik}(K_{ik} + K_{i,k+1})]T_{ik}^{m+1} - \eta_{ik} K_{i,k+1} T_{i,k+1}^{m+1} = T_{ik}^m + \eta_{ik} Q_k, \tag{74}$$

$$\begin{aligned}
-\eta_{iN}(K_{i,N} - \gamma_2 K_{i,N+1})T_{i,N-1}^{m+1} + [1 + \eta_{iN}(K_{iN} + \gamma_1 K_{i,N+1})]T_{iN}^{m+1} = \\
\eta_{iN} K_{i,N+1}(\gamma_1 + \gamma_2)T_f + T_{iN}^m + \eta_{iN} Q_N,
\end{aligned} \tag{75}$$

where $\eta_{ik} \equiv \Delta t/(\rho_i c_{ik} \Delta h_i)$.

Next consider case 2 ($T_{sf} = 0°\mathrm{C}$ and snow present). Since T_{sf} is fixed, there is no surface flux equation. The new snow temperature is given by (71), but with the unknown T_{sf}^{m+1} replaced by $T_{sf} = 0°\mathrm{C}$. In matrix form we have

$$[1 + \eta_s(K_{int} + K_s)]T_s^{m+1} - \eta_s K_{int} T_{i1}^{m+1} = \eta_s K_s T_{sf} + T_s^m. \tag{76}$$

The ice equations for case 2 are the same as for case 1: (73), (74), and (75).

For case 3 ($T_{sf} < 0°\mathrm{C}$ and snow absent) the surface temperature equation is of the form (69), but we use a second-order accurate expression for dT/dz at $z = 0$:

$$F_0^* + \left(\frac{dF_0}{dT_{sf}}\right)^* (T_{sf}^{m+1} - T_{sf}^*) = K_{i1}[\gamma_1(T_{sf}^{m+1} - T_{i1}^{m+1}) + \gamma_2(T_{sf}^{m+1} - T_{i2}^{m+1})].$$

This gives the matrix equation

$$\left[\left(\frac{dF_0}{dT_{sf}}\right)^* - K_{i1}(\gamma_1 + \gamma_2)\right] T_{sf}^{m+1} + \gamma_1 K_{i1} T_s^{m+1} + \gamma_2 K_{i1} T_{i2}^{m+1} = \left(\frac{dF_0}{dT_{sf}}\right)^* T_{sf}^* - F_0^*. \tag{77}$$

The equation for T_{i1} in case 3 is

$$\rho_i c_{i1} \frac{(T_{i1}^{m+1} - T_{i1}^m)}{\Delta t} = \frac{1}{\Delta h_i} K_{i1}[\gamma_1(T_{sf}^{m+1} - T_{i1}^{m+1}) + \gamma_2(T_{sf}^{m+1} - T_{i2}^{m+1})] - K_{i2}(T_{i1}^{m+1} - T_{i2}^{m+1}) + Q_1.$$

Rearranging terms, we find

$$-\eta_{i1} K_{i1}(\gamma_1 + \gamma_2) T_{sf}^{m+1} + [1 + \eta_{i1}(K_{i2} + \gamma_1 K_{i1})] T_{i1}^{m+1} - \eta_{i1}(K_{i2} - \gamma_2 K_{i1}) T_{i2}^{m+1} = T_{i1}^m + \eta_{i1} Q_1. \quad (78)$$

Equation (77) includes T_{i2}^{m+1} and therefore gives an unwanted matrix term two places to the right of the main diagonal. We eliminate this term by making the substitution

$$R_1 \rightarrow c_2 R_1 - c_1 R_2,$$

where R_1 is the first matrix row, R_2 is the second row, and $c_1 = \gamma_2 K_{i1}$ and $c_2 = -\eta_{i1}(K_{i2} - \gamma_2 K_{i1})$ are the coefficients multiplying T_{i2}^{m+1} in rows 1 and 2, respectively. The other ice layer equations for case 3 are (74) and (75).

Finally, for case 4 ($T_{sf} = 0°\mathrm{C}$ and snow absent) we have the top ice layer equation

$$\rho_i c_{i1} \frac{(T_{i1}^{m+1} - T_{i1}^m)}{\Delta t} = \frac{1}{\Delta h_i} K_{i1}[\gamma_1(T_{sf} - T_{i1}^{m+1}) + \gamma_2(T_{sf} - T_{i2}^{m+1})] - K_{i2}(T_{i1}^{m+1} - T_{i2}^{m+1}) + Q_1.$$

which can be rewritten as

$$[1 + \eta_{i1}(\gamma_1 K_{i1} + K_{i2})] T_{i1}^{m+1} + \eta_{i1}(\gamma_2 K_{i1} - K_{i2}) T_{i2}^{m+1} = \eta_{i1} K_{i1}(\gamma_1 + \gamma_2) T_{sf} + T_{i1}^m + \eta_{i1} Q_1. \quad (79)$$

The remaining ice layer equations are (74) and (75), as with the other three cases.

This completes the specification of the matrix equations for the four cases. We compute the new temperatures using a tridiagonal solver. After each iteration we check to see whether the following conditions hold:

1. $T_{sf} \leq 0°\mathrm{C}$.
2. The change in T_{sf} since the previous iteration is less than a prescribed limit, $\Delta T_{\max}$.
3. $F_0 \geq F_{ct}$. (If $F_0 < F_{ct}$, ice would be growing at the top surface, which is not allowed.)
4. The rate at which energy is added to the ice by the external fluxes equals the rate at which the internal ice energy is changing, to within a prescribed limit $\Delta F_{\max}$.

We also check the convergence rate of T_{sf}. If T_{sf} is oscillating and failing to converge, we average temperatures from successive iterations to improve convergence. When all these conditions are satisfied—usually within two to four iterations for $\Delta T_{\max} \approx 0.01°\mathrm{C}$ and $\Delta F_{max} \approx 0.01\ \mathrm{W/m^2}$—the calculation is complete.

3.5.3 Growth and melting

We first derive expressions for the enthalpy q. The enthalpy of snow (or fresh ice) is given by

$$q_s(T) = -\rho_s(-c_0 T + L_0).$$

Sea ice enthalpy is more complex, because of brine pockets whose salinity varies inversely with temperature. The specific heat of sea ice, given by (64), includes not only the energy needed to warm or cool ice, but also

the energy used to freeze or melt ice adjacent to brine pockets. Equation (64) can be integrated to give the energy δ_e required to raise the temperature of a unit mass of sea ice of salinity S from T to T':

$$\delta e(T, T') = c_0(T' - T) + L_0 \mu S \left(\frac{1}{T} - \frac{1}{T'} \right).$$

If we let $T' = T_m \equiv -\mu S$, the temperature at which the ice is completely melted, we have

$$\delta e(T, T_m) = c_0(T_m - T) + L_0 \left(1 - \frac{T_m}{T} \right).$$

Multiplying by ρ_i to change the units from $\mathrm{J/kg}$ to $\mathrm{J/m^3}$ and adding a term for the energy needed to raise the meltwater temperature to $0^\circ\mathrm{C}$, we obtain the sea ice enthalpy:

$$q_i(T, S) = -\rho_i \left[c_0(T_m - T) + L_0 \left(1 - \frac{T_m}{T} \right) - c_w T_m. \right] \tag{80}$$

Note that (80) is a quadratic equation in T. Given the layer enthalpies we can compute the temperatures using the quadratic formula:

$$T = \frac{-b - \sqrt{b^2 - 4ac}}{2a},$$

where

$$\begin{aligned} a &= c_0, \\ b &= (c_w - c_0)\, T_m - \frac{q_i}{\rho_i} - L_0, \\ c &= L_0 T_m. \end{aligned}$$

The other root is unphysical.

Melting at the top surface is given by

$$q\, \delta h = \begin{cases} (F_0 - F_{ct})\, \Delta t & \text{if } F_0 > F_{ct} \\ 0 & \text{otherwise} \end{cases} \tag{81}$$

where q is the enthalpy of the surface ice or snow layer (recall that $q < 0$) and δh is the change in thickness. If the layer melts completely, the remaining flux is used to melt the layers beneath. Any energy left over when the ice and snow are gone is added to the ocean mixed layer. Ice cannot grow at the top surface, but new snow can fall. Snowfall is added at the end of the thermodynamic time step.

Growth and melting at the bottom ice surface are governed by

$$q\, \delta h = (F_{cb} - F_{bot})\, \Delta t, \tag{82}$$

where F_{bot} is given by (61) and F_{cb} is the conductive heat flux at the bottom surface:

$$F_{cb} = \frac{k_{i,N+1}}{\Delta h_i} [\gamma_1 (T_{iN} - T_f) + \gamma_2 (T_{i,N-1} - T_f)].$$

If ice is melting at the bottom surface, q in (82) is the enthalpy of the bottom ice layer. If ice is growing, q is the enthalpy of new ice with temperature T_f and salinity S_{max}. This ice is added to the bottom layer.

If the latent heat flux is negative (i.e., latent heat is transferred from the ice to the atmosphere), snow or snow-free ice sublimates at the top surface. If the latent heat flux is positive, vapor from the atmosphere is deposited at the surface as snow or ice. The thickness change of the surface layer is given by

$$(\rho L_v - q)\delta h = F_l \Delta t, \tag{83}$$

where ρ is the density of the surface material (snow or ice), and $L_v = 2.501 \times 10^6$ J/kg is the latent heat of vaporization of liquid water at 0°C. Note that ρL_v is nearly an order of magnitude larger than typical values of q. For positive latent heat fluxes, the deposited snow or ice is assumed to have the same enthalpy as the existing surface layer.

After growth and melting, the various ice layers no longer have equal thicknesses. We therefore adjust the layer interfaces, conserving energy, so as to restore layers of equal thickness $\Delta h_i = h_i/N_i$. This is done by computing the overlap η_{km} of each new layer k with each old layer m:

$$\eta_{km} = \min(z_m, z_k) - \max(z_{m-1}, z_{k-1}),$$

where z_m and z_k are the vertical coordinates of the old and new layers, respectively. The enthalpies of the new layers are

$$q_k = \frac{1}{\Delta h_i} \sum_{m=1}^{N_i} \eta_{km} q_m.$$

At the end of the time step we check whether the snow is deep enough to lie partially below freeboard (i.e., below the surface of the ocean). From Archimedes' principle, the base of the snow is at freeboard when

$$\rho_i h_i + \rho_s h_s = \rho_w h_i.$$

Thus the snow base lies below freeboard when

$$h^* \equiv h_s - \frac{(\rho_w - \rho_i) h_i}{\rho_s} > 0.$$

In this case we raise the snow base to freeboard by converting some snow to ice:

$$\begin{aligned} \delta h_s &= \frac{-\rho_i h^*}{\rho_w}, \\ \delta h_i &= \frac{\rho_s h^*}{\rho_w}. \end{aligned}$$

In rare cases this process can increase the ice thickness substantially. For this reason we postpone snow-ice conversions until after the remapping in thickness space (Section 3.2), which assumes that ice growth during a single time step is fairly small.

Lateral melting is accomplished by multiplying the state variables by $1 - r_{side}$, where r_{side} is the fraction of ice melted laterally, and adjusting the ice energy and fluxes as appropriate.

4 Numerical implementation

CICE is written in fixed-format FORTRAN90 and runs on UNIX host platforms. The code is parallelized via grid decomposition with MPI and has been optimized for vector architectures.

A second, "external" layer of parallelization involves message passing between CICE and the flux coupler, which may be running on different machines in a distributed system. The parallelization scheme for CICE was designed so that MPI could be used for the coupling along with MPI or no parallelization internally. The internal parallelization method is set at compile time with the `BINTYPE` definition in the make scripts for the stand-alone model. Message passing between the ice model and the flux coupler is accomplished with MPI, regardless of the type of internal parallelization used for CICE.

4.1 Directory structure

The present code distribution includes make files, several scripts and some input files. The main directory is **cice/**, and a run directory (**rundir**) is created upon initial execution of the script **comp_ice**. One year of atmospheric forcing data is also available from the code distribution web site (see the **README** file for details).

cice/

README_v3.14 basic information

bld/ makefiles

Macros.⟨OS⟩.⟨SITE⟩.⟨machine⟩ macro definitions for the given operating system, used by **Makefile.**⟨OS⟩

Makefile.⟨OS⟩ primary makefile for the given operating system (⟨std⟩ works for most systems)

makedep.c perl script that determines module dependencies

clean_ ice script that removes files from the compile directory

comp_ ice script that sets up the run directory and compiles the code

doc/ documentation

cicedoc.pdf this document

PDF/ PDF documents of numerous publications related to CICE

ice.log.⟨OS⟩.⟨SITE⟩.⟨machine⟩ sample diagnostic output files

input_templates/ input files that may be modified for other CICE configurations

global_gx1.grid ⟨1°⟩ displaced pole grid

global_gx1.kmt ⟨1°⟩ land mask

global_gx3.grid ⟨3°⟩ displaced pole grid

global_gx3.kmt ⟨3°⟩ land mask

ice.restart_file pointer for restart file name

ice_in namelist input data (data paths depend on particular system)

iced_gx3_v3.1 restart file used for initial condition

iced_gx1_v3.14 restart file used for initial condition

run_ice.⟨OS⟩.⟨SITE⟩.⟨machine⟩ sample script for running on the given operating system

source/ CICE source code

CICE.F main program

CICE.F_debug debugging version of **CICE.F**

ice_albedo.F albedo parameterization

ice_atmo.F stability-based parameterization for calculation of turbulent ice-atmosphere fluxes

ice_calendar.F keeps track of what time it is

ice_constants.F physical and numerical constants and parameters

ice_coupling.F interface with the flux coupler
ice_diagnostics.F miscellaneous diagnostic and debugging routines
ice_domain.F MPI subdomain sizes and related parallel processing info
ice_dyn_evp.F elastic-viscous-plastic dynamics component
ice_exit.F aborts the model, printing an error message
ice_fileunits.F unit numbers for I/O
ice_flux.F fluxes needed/produced by the model
ice_flux_in.F routines to read and interpolate forcing data for stand-alone ice model runs
ice_grid.F grid and land masks
ice_history.F netCDF output routines and restart read/write
ice_init.F namelist and initializations
ice_itd.F utilities for managing ice thickness distribution
ice_itd_linear.F linear remapping for transport in thickness space
ice_kinds_mod.F basic definitions of reals, integers, etc.
ice_mechred.F mechanical redistribution component (ridging)
ice_model_size.F grid size and number of thickness categories and vertical layers
ice_model_size.F.gx1 specific ice_model_size.F for use by scripts with $\langle 1^o \rangle$ grid
ice_model_size.F.gx3 specific ice_model_size.F for use by scripts with $\langle 3^o \rangle$ grid
ice_mpi_internal.F utilities for internal MPI parallelization
ice_ocean.F mixed layer ocean model
ice_read_write.F utilities for reading and writing files
ice_scaling.F ice-area scaling of variables for the coupler
ice_state.F essential arrays to describe the state of the ice
ice_therm_itd.F thermodynamic changes mostly related to ice thickness distribution (post-coupling)
ice_therm_vertical.F vertical growth rates and fluxes (pre-coupling thermodynamics)
ice_timers.F timing routines
ice_transport_mpdata.F horizontal advection via MPDATA or upwind
ice_transport_remap.F horizontal advection via incremental remapping
ice_work.F globally accessible work arrays

rundir/ execution or "run" directory created when the code is compiled using the **comp_ice** script

cice code executable
compile/ directory containing object files, etc.
grid horizontal grid file from **cice/input_templates/**
ice.log.[ID] diagnostic output file
ice_in namelist input data from **cice/input_templates/**
hist/iceh_mavg.[timeID].nc monthly average output history file
kmt land mask file from **cice/input_templates/**
restart/ restart directory
iced_gx3_v3.1 initial condition from **cice/input_templates/**
ice.restart_file restart pointer from **cice/input_templates/**
run_ice batch run script file from **cice/input_templates/**

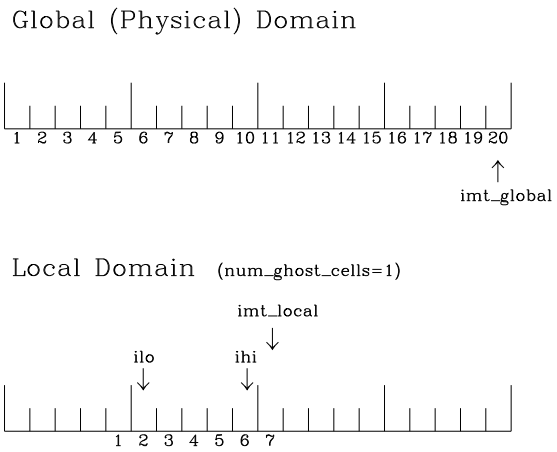

Figure 6: Grid parameters for a sample one-dimensional, 20-cell global domain decomposed into four local subdomains. Each local domain has one ghost cell on each side, and the physical portion of the local domains are labeled `ilo:ihi`. The parameter `imt_local` is the total number of cells in the local domain, including ghost cells, and the same numbering system is applied to each of the four subdomains.

4.2 Grid, boundary conditions and masks

The spatial discretization is specialized for a generalized orthogonal B-grid as in [33] or [28]. The ice and snow area, volume and energy are given at the center of the cell, velocity is defined at the corners, and the internal ice stress tensor takes four different values within a grid cell; bilinear approximations are used for the stress tensor and the ice velocity across the cell, as described in [15]. This tends to avoid the grid decoupling problems associated with the B-grid.

Since ice thickness and thermodynamic variables such as temperature are given in the center of each cell, the grid cells are referred to as "T cells." We also occasionally refer to "U cells," which are centered on the northeast corner of the corresponding T cells and have velocity in the center of each. The velocity components are aligned along grid lines.

In general, the global gridded domain is `imt_global` x `jmt_global`, while the subdomains used in the MPI grid decomposition are `imt_local` x `jmt_local`. The physical portion of a subdomain is indexed as [`ilo:ihi,jlo:jhi`], with `num_ghost_cells` "ghost" cells outside the domain, used for boundary conditions. These parameters are illustrated in Figure 6 in one dimension. The routines *global_scatter* and *global_gather* distribute information from the global domain to the local domains and back, respectively. If MPI is not being used for grid decomposition in the ice model, these routines simply adjust the indexing on the global domain to the single, local domain index coordinates. We strongly suggest that the user choose the number of local domains so that the global domain is evenly divided. If the global domain is not evenly divided by the number of processors, then the last subdomain will contain nonphysical points ("padding"). Besides a loss of efficiency due to computing at these points, other problems may arise due to incompatible initializations and spurious data values.

The user has three choices of grid routines: *popgrid* reads grid lengths and other parameters for a nonuniform grid, *rectgrid* creates a regular rectangular grid, and *columngrid* creates a column model configuration (`imt_global` and `jmt_global` are both 1). The input files **global_gx3.grid** and **global_gx3.kmt** contain the $\langle 3^\circ \rangle$ POP grid and land mask; **global_gx1.grid** and **global_gx1.kmt** contain the $\langle 1^\circ \rangle$ grid and land mask. These are binary unformatted, direct access files produced on an SGI (Big Endian). If you are using an incompatible (Little Endian) architecture, choose `rectangular` instead of `displaced_pole` in **ice_in**, or follow procedures as for the SGI Altix ($\langle$**OS**$\rangle$.$\langle$**SITE**$\rangle$.$\langle$**machine**$\rangle$= `Linux.LANL.mauve`).

In the current implementation with a bipolar, displaced-pole grid, at least one row of grid cells along the north and south boundaries are assumed to be located on land. Along domain boundaries not masked by land, periodic conditions wrap the domain around the globe. The original boundary routine is *bound*; the other boundary routines improve parallel performace by not filling all four boundaries when that is unnecessary, and by updating multiple spatial arrays at once. The boundary routines also perform boundary communications between local domains when MPI is in use.

A land mask `hm` (M_h) is specified in the cell centers, with 0 representing land and 1 representing ocean cells. A corresponding mask `uvm` (M_u) for velocity and other corner quantities is given by

$$M_u(i,j) = \min\{M_h(l),\, l = (i,j),\, (i+1,j),\, (i,j+1),\, (i+1,j+1)\}.$$

The logical masks `tmask` and `umask` (which correspond to the real masks `hm` and `uvm`, respectively) are useful in conditional statements.

In addition to the land masks, two other masks are implemented in *evp_prep* in order to reduce the dynamics component's work on a global grid. At each time step the logical masks `ice_tmask` and `ice_umask` are determined from the current ice extent, such that they have the value "true" wherever ice exists. They also include a border of cells around the ice pack for numerical purposes. These masks are used in the dynamics component to prevent unnecessary calculations on grid points where there is no ice. They are not used in the thermodynamics component, so that ice may form in previously ice-free cells. Like the land masks `hm` and `uvm`, the ice extent masks `ice_tmask` and `ice_umask` are for T cells and U cells, respectively.

Two additional masks are created for the user's convenience: `mask_n` and `mask_s` can be used to compute or write data only for the northern or southern hemispheres, respectively.

4.3 Initialization and coupling

The ice model's parameters and variables are initialized in several steps. Many constants and physical parameters are set in **ice_constants.F**. Namelist variables (Table 4), whose values can be altered at run time, are handled in *input_data*. These variables are given default values in the code, which may then be changed when the input file **ice_in** is read. Several variables available in the namelist declaration given in **ice_init.F** are not usefully implemented in the current version of CICE; these variables are used in the NCAR CCSM ice model and are included in the namelist declaration for consistency with that code. Physical constants, numerical parameters, and variables are first set in initialization routines for each ice model component or module. Then, if the ice model is being restarted from a previous run, some variables are read and reinitialized in *restartfile*. Finally, albedo is initialized based on the initial ice state. Some of these parameters will be described in more detail in Table 4.

The ice component communicates with the flux coupler by passing messages using MPI, which is initialized in *setup_mpi* for both coupled and stand-alone MPI runs. Further initialization for coupling occurs in *ice_coupling_setup* and *init_cpl*. The routines *from_coupler* and *to_coupler* respectively unpack and pack the data being passed between the ice component and the flux coupler, and perform necessary averages and unit conversions.

For stand-alone runs, routines in **ice_flux_in.F** read and interpolate data from files, and are intended merely to provide guidance for the user to write his or her own routines. Whether the code is to be run in stand-alone or coupled mode is determined at compile time, as described below.

4.4 Choosing an appropriate time step

The time step is chosen based on stability of the transport component (both horizontal and in thickness space) and on resolution of the physical forcing. CICE allows the dynamics, advection and ridging portion

of the code to be run with a shorter timestep, Δt_{dyn} (`dt_dyn`), than the thermodynamics timestep Δt (`dt`). In this case, `dt` and the integer `ndyn_dt` are specified, and `dt_dyn = dt/ndyn_dt`.

A conservative estimate of the horizontal transport time step bound, or CFL condition, under remapping yields

$$\Delta t_{dyn} < \frac{\min(\Delta x, \Delta y)}{2\max(u, v)}.$$

Note that this is a factor of 2 larger than the MPDATA condition. Numerical estimates for this bound for several POP grids, assuming $\max(u, v) = 0.5$ m/s, are as follows:

grid label	N pole singularity	dimensions	$\min\sqrt{\Delta x \cdot \Delta y}$	max Δt_{dyn}
gx3	Greenland	100×116	39×10^3 m	10.8 hr
gx1	Greenland	320×384	18×10^3 m	5.0 hr
p4	Canada	900×600	6.5×10^3 m	1.8 hr

As discussed in section 3.3 and [23], the maximum time step in practice is usually determined by the time scale for large changes in the ice strength (which depends in part on wind strength). Using the strength parameterization of [30], as in eq. 50 , limits the time step to 30 minutes for the old ridging scheme, and to 2 hours for the new scheme, assuming Δx = 10 km. Practical limits may be somewhat less, depending on the strength of the atmospheric winds.

Transport in thickness space imposes a similar restraint on the time step, given by the ice growth/melt rate and the smallest range of thickness among the categories, $\Delta t < \min \Delta H / 2 \max f$, where ΔH is the distance between category boundaries and f is the thermodynamic growth rate. For the 5-category ice thickness distribution used as the default in this distribution, this is not a stringent limitation: $\Delta t < 19.4$ hr, assuming $\max f = 40$ cm/day.

The dynamics component is subcycled `ndte` (N) times per time step so that the elastic waves essentially disappear before the next time step. The subcycling time step (Δt_e) is thus

```
dte = dt_dyn/ndte.
```

A second parameter, $E_\circ$ (`eyc`), must be selected, which defines the elastic wave damping timescale T, described in Section 3.4, as `eyc*dt_dyn`. The forcing terms are not updated during the subcycling. Given the small step (`dte`) at which the EVP dynamics model is subcycled, the elastic parameter E is also limited by stability constraints, as discussed in [14]. Linear stability analysis for the dynamics component shows that the numerical method is stable as long as the subcycling time step Δt_e sufficiently resolves the damping timescale T. For the stability analysis we had to make several simplifications of the problem; hence the location of the boundary between stable and unstable regions is merely an estimate. In practice, the ratio $\Delta t_e : T : \Delta t$ = 1 : 40 : 120 provides both stability and acceptable efficiency for time steps (Δt) on the order of 1 hour.

Note that only T and Δt_e figure into the stability of the dynamics component; Δt does not. The thermodynamics component is stable for any time step. Although the time step may not be tightly limited by stability considerations, large time steps (*eg.,* $\Delta t = 1$ day, given daily forcing) do not produce accurate results in the dynamics component. The reasons for this error are discussed in [14]; see [17] for its practical effects.

4.5 Model output

Model output data is averaged over the period given by `histfreq` and written to netCDF files prepended by `history_file` in **ice_in**. That is, if `history_file='iceh'` then the filenames will have the form **iceh_[timeID].nc**. Header information for data contained in these files is displayed with the command

`ncdump -h filename.nc`. With this release, standard ice data fields are output. The user may add (or subtract) variables not already available in the namelist by following the instructions in **ice_history.F**.

A few thermodynamic variables have special `_hist` forms in addition to the standard quantity used in the code. These are variables that are initialized in the middle of the time step (at the beginning of the second thermodynamics routine, *thermo_itd*), just after being sent to the coupler, although they may change at the beginning of the time step (in *thermo_vertical*). The "standard" variable initialized as such contains a full time step's worth of data when it is sent to the coupler; its history counterpart is initialized at the beginning of the time step and so also contains a full time step's worth of data, although its value may be slightly different from that sent to the coupler. This code modification was made for coupled model load balancing.

The normalized principal components of internal ice stress are computed in *principal_stress* and written to the history file. This calculation is not necessary for the simulation; principal stresses are merely computed for diagnostic purposes and included here for the user's convenience.

Like `histfreq`, the parameter `diagfreq` can be used to regulate how often output is written. In the present code, `diagfreq` is used to determine the frequency with which diagnostic data are written to the log file. The log file unit to which diagnostic output is written is set in **ice_fileunits.F**. If `diag_type = 'stdout'`, then it is written to standard out (or to **ice.log.[ID]** if you redirect standard out as in **run_ice**); otherwise it is written to the file given by `diag_file`. In addition to the standard diagnostic output (maximum area-averaged thickness, velocity, average albedo, total ice area, and total ice and snow volumes), the namelist options `print_points` and `print_global` cause additional diagnostic information to be computed and written. `print_global` outputs global sums that are useful for checking global conservation of mass and energy. `print_points` writes data for two specific grid points. Currently, one point is near the North Pole and the other is in the Weddell Sea; these may be changed in **ice_diagnostics.F**.

A binary unformatted file is created that contains all of the data that CICE needs for a full restart. The filename begins with the character string `dumpfile`, and the restart dump frequency is given by `dumpfreq` and `dumpfreq_n`. The pointer to the filename from which the restart data is to be read is set in `pointer_file`.

Timing routines are included in **ice_timers.F**. To use the timers, first initialize them with *ice_timer_clear*, then wrap the segment of code you would like to time with *ice_timer_start* and *ice_timer_stop*. Finally, *ice_timer_print* writes the results to the log file. Each of these routines takes a single argument, the timer number. Calling *ice_timer_clear* or *ice_timer_print* with an argument of -1 initializes all of the timers at once, or prints all of the timings, rather than having to call each individually. Currently, the timers are set up as in Table 3.

The timings provided by these timers are not mutually exclusive. For example, the column timer (4) includes the timings from 5, 6 and 7, and subroutine *bound* (timer 10) is called from many different places in the code, including the dynamics and advection routines.

The timers use *MPI_WTIME* for parallel runs and the F90 intrinsic *system_clock* for single-processor runs.

4.6 Execution procedures

To compile and execute the code: in the source directory,

1. Alter directories in the script **comp_ice**,
2. Run **comp_ice** to set up the run directory and make the executable '**cice**,'
3. To clean the compile directory and start fresh, alter the script **clean_ice** and execute it.

In the run directory,

Timer Number	Label	
0	Total	the entire run
1	TimeLoop	Total minus initialization and exit
2	Dynamics	EVP
3	Advectn	horizontal transport
4	Column	all vertical (column) processes
5	Thermo	vertical thermodynamics
6	Ridging	mechanical redistribution
7	Cat Conv	transport in thickness space
8	Coupling	sending/receiving coupler messages
9	ReadWrit	reading/writing files
10	Bound	boundary conditions and subdomain communications

Table 3: CICE timers.

1. Alter `atm_data_dir` and `ocn_data_dir` in the namelist file **ice_in**,
2. Alter the script **run_ice** for your system,
3. Execute **run_ice**.

If this fails, see Section 5.1.

This procedure creates the output log file **ice.log.[ID]**, and if `npt` is long enough compared with `dumpfreq` and `histfreq`, dump files **iced.[timeID]** and netCDF history output files **iceh_[timeID].nc**. Using the $\langle 3^\circ \rangle$ grid, the log file should be similar to **ice.log.**$\langle$**OS**$\rangle$, provided for the user's convenience. These log files were created using MPI on 8 processors (`NX=4` and `NY=2`), on the $\langle 3^\circ \rangle$ grid.

Several precompiler options are available in **comp_ice** for configuring the run:

location	variable	options	description
comp_ice	RES	gx3, gx1	grid resolution
	BINTYPE	MPI	use MPI for internal parallelization
	NX, NY	(integers)	number of MPI processors assigned to each coordinate direction (NY$\leq$ 2)

The scripts define a number of environment variables, mostly as directories that you will need to edit for your own environment. Two of these environment variables are defined externally, `$HOME`, which points to your home directory (where we assume the CICE directory is installed), and `$SYSTEM_USERDIR`, which points to scratch disks on the machines at Oak Ridge National Laboratory.

CICE namelist variables available for changes after compile time appear in **ice.log.*** with values read from the file **ice_in**; their definitions are given in Section 5.4. For example, to run for a different length of time, say three days, set `npt=72` in **ice_in**. At present, the user supplies the time step `dt`, the number of dynamics/advection/ridging subcycles `ndyn_dt`, and the number of evp subcycles `ndte`, and `dte` is then calculated in subroutine *init_evp*. The primary reason for doing it this way is to ensure that `ndte` is an integer.

To restart from a previous run, set the filename in **ice.restart_file** (created by the previous run) to the desired data file (**iced.[timeID]**), then set `restart=.true.` in **ice_in**. Restarts are exact for MPI or single processor runs.

The structure and flow of the sea ice code are fairly well outlined in the main driver routine **CICE.F**. Note that the thermodynamics routine is broken into two pieces, so that fluxes can be returned to the coupler as quickly as possible. This enables the flux coupler to deliver the thermodynamic fluxes to other component models while the ice model continues running.

5 Troubleshooting

5.1 Initial setup

The scrip **comp_ice** is configured so that the files **grid**, **kmt**, **ice_in**, **run_ice**, **iced_gx3_v3.1** and **ice.restart_file** are NOT overwritten after the first setup. If you wish to make changes to the original files in **input_templates/** rather than those in the run directory, either remove the files from the run directory before executing **comp_ice** or edit the script.

If the code fails to compile or run, or if the model configuration is changed, try the following:

- create **Macros.***. **Makefile.*** and **run_ice.*** files for your particular platform, if they do not already exist (type 'uname -s' at the prompt and compare the result with the file suffixes; we rename `UNICOS/mp` as `UNICOS` for simplicity)
- modify the `INCLUDE` directory path and other settings for your system in the scripts, **Macros.*** and **Makefile.*** files.
- alter directory paths, file names and the execution command as needed in **run_ice** and **ice_in**.
- set the internal parallelization method and number of processors (`BINTYPE, NX, NY`) in **Macros.***. `NX` and `NY` should evenly divide the respective number of grid points in each direction. We suggest that `NY`≤ 2 for load balancing; otherwise processors assigned subdomains near the equator have little work to do (we hope).
- for stand-alone runs, check that `-Dcoupled` is *not* set in the **Macros.*** file.
- for coupled runs, check that `-Dcoupled` and `-DCCSMcoupled` are set in the **Macros.*** file. You may compile the model as above or use NCAR scripts that set up and compile all of the CCSM coupled model components at once (not available in this distribution), using `-DCCSMcoupled`. The option `-Dfcd_coupled` controls another model configuration not available in this distribution.
- edit the grid size and other parameters in **source/ice_model_size.F**.

5.2 Slow execution

On some architectures, underflows (10^{-300} for example) are not flushed to zero automatically. Usually a compiler flag is available to do this, but if not, try uncommenting the block of code at the end of subroutine *stress* in **ice_dyn_evp.F**. You will take a hit for the extra computations, but it will not be as bad as running with the underflows.

5.3 Debugging hints

Several utilities are available that can be helpful when debugging the code. Not all of these will work everywhere in the code, due to possible conflicts in module dependencies.

debug_ice **(CICE.F)** A wrapper for *print_state* that is easily called from numerous points during the timestepping loop (see **CICE.F_debug**).

print_state **(ice_diagnostics.F)** Print the ice state and forcing fields for a given grid cell.

ice_global_real_minmax **(ice_mpi_internal.F)** Compute and print the minimum and maximum values for a real array. A sister routine prints the global sum of all elements in a real array, *ice_global_real_sum*.

`diag = .true.` **(**in calls to *ice_read***)** Print global max and min values for the field being read.

`dbug = .true.` **(ice_in)** Print numerous diagnostic quantities for forcing data read in **ice_flux_in.F**.

`print_global` **(ice_in)** If true, compute and print numerous global sums for energy and mass balance analysis. This option can significantly degrade code efficiency.

`print_points` **(ice_in)** If true, print numerous diagnostic quantities for two grid cells, one near the north pole and one in the Weddell Sea. This utility also provides the local grid indices and processor number (`ip, jp, my_task`) for these points, which can be used in conjunction with `check_step`, to call *print_state*. These flags are set in **ice_diagnostics.F**. This option can be fairly slow, due to gathering data from MPI subdomains.

5.4 Known bugs

1. Fluxes sent to the coupler may have incorrect values in grid cells that change from an ice-free state to having ice during the given time step, or vice versa, due to scaling by the ice area. The authors of the flux coupler insist on the area scaling so that the ice and land models are treated consistently in the coupler (but note that the land area does not suddenly become zero in a grid cell, as does the ice area).

2. A sizable fraction (more than 10%) of the total shortwave radiation is absorbed at the surface but should be penetrating into the ice interior instead. This is due to use of the aggregated, effective albedo rather than the bare ice albedo when `snowpatch` < 1, and fixing the problem will require more albedo arrays to be added to the code.

3. The date-of-onset diagnostic variables, `melt_onset` and `frz_onset`, are not included in the restart file, and therefore may be incorrect for the current year if the run is restarted after Jan 1. Also, these variables were implemented with the Arctic in mind and may be incorrect for the Antarctic.

4. The single-processor *system_clock* time may give erratic results on some architectures.

5. Local domains are not padded for uneven division of the global domain.

6. History files that contain time averaged data (`hist_avg = .true.` in **ice_in**) will be incorrect if restarting from midway through an averaging period.

7. In stand-alone runs, restarts from the end of `ycycle` will not be exact.

Acknowledgments and Copyright

This work has been supported through the Department of Energy Computer Hardware Applied Mathematics and Model Physics (CHAMMP) program, Climate Change Prediction Program (CCPP), and Scientific Discovery through Advanced Computing (SCIDAC) program, with additional support from the T-3 Fluid Dynamics Group at Los Alamos National Laboratory. Special thanks are due to the following people:

- members of the CCSM Polar Climate Working Group, including David Bailey Cecilia Bitz, Bruce Briegleb, Tony Craig, Marika Holland, and Julie Schramm.

- Clifford Chen of Fujitsu, who spent long hours testing and improving the model's vector performance.
- Trey White of Oak Ridge National Laboratory, and Wieslaw Maslowski and colleagues at the Naval Postgraduate School, for assistance porting the code to the X1.
- the many researchers who tested beta versions of CICE 3.14 and waited patiently for the official release.

Table of namelist options

variable	options/format	description	recommended value
albicei	$0 < \alpha < 1$	near infrared ice albedo for thicker ice	
albicev	$0 < \alpha < 1$	visible ice albedo for thicker ice	
albsnowi	$0 < \alpha < 1$	near infrared, cold snow albedo	
albsnowv	$0 < \alpha < 1$	visible, cold snow albedo	
advection	remap	linear remapping advection	'remap'
	mpdata	2nd order MPDATA	
	upwind	1st order MPDATA	
atm_data_dir	path/	path to atmospheric forcing data directory	
atm_data_type	default	constant values defined in the code	
	ncar	NCAR bulk forcing data	
	LYq	AOMIP/Large-Yeager forcing data	
dbug	true/false	if true, write atm/ocn data diagnostics	.false.
diag_file	filename	diagnostic output file	
diag_type	stdout	write diagnostic output to stdout	'stdout' (if uncoupled)
	file	write diagnostic output to file	
diagfreq	integer	frequency of diagnostic output in dt	24
	eg., 10	once every 10 time steps	
dt	seconds	thermo/transport time step length	3600.
dump_file	filename prefix	output file for restart dump	'iced'
dumpfreq	y	write restart every dumpfreq_n years	'y'
	m	write restart every dumpfreq_n months	
	d	write restart every dumpfreq_n days	
dumpfreq_n	integer	frequency restart data is written	
evp_damping	true/false	if true, damp elastic waves [13]	.false.
fyear_init	yyyy	first year of atmospheric forcing data	
f_$\langle var \rangle$	true/false	write $\langle var \rangle$ to history	

Table 4: Namelist options (continued next page).

variable	options/format	description	recommended value
grid_file	filename	name of grid file to be read	'grid'
grid_type	rectangular	defined in *rectgrid*	'displaced_pole'
	displaced_pole	read from file in *popgrid*	
hist_avg	true	write time-averaged data	.true.
	false	write snapshots of data	
hist_dir	path/	path to history output directory	
histfreq	y	write history output once a year	'm'
	m	write history output once a month	
	w	write history output once a week	
	d	write history output once a day	
	1	write history output every time step	
history_file	filename prefix	output file for history	'iceh'
ice_ic	default	latitude and sst dependent	'default'
	none	no ice	
istep0	integer	initial time step number	0
kcatbound	0	original category boundary formula	0
	1	new category boundary formula	
kdyn	0	dynamics OFF	1
	1	EVP dynamics	
kitd	0	delta function ITD approximation	1
	1	linear remapping ITD approximation	
kmt_file	filename	name of land mask file to be read	'kmt'
krdg_partic	0	old ridging participation function	1
	1	new ridging participation function	
krdg_redist	0	old ridging redistribution function	1
	1	new ridging redistribution function	
kstrength	0	ice strength formulation [11]	1
	1	ice strength formulation [30]	
ndte	integer	number of EVP subcycles	120
ndyn_dt	integer	number of dynamics/advection/ridging steps per thermo timestep	1
npt	integer	total number of time steps to take	
oceanmixed_file	filename	data file containing ocean forcing data	
oceanmixed_ice	true/false	active ocean mixed layer calculation	.true. (if uncoupled)
ocn_data_dir	path/	path to oceanic forcing data directory	
print_points	true/false	print diagnostic data for two grid points	.false.
precip_units	mm_per_month	liquid precipitation data units	
	mm_per_sec	(default; MKS units)	
print_global	true/false	print diagnostic data, global sums	.false.
restart	true/false	initialize using restart file	.true.
restart_dir	path/	path to restart directory	
restore_sst	true/false	restore sst to data	

Table 5: Namelist options (continued next page).

variable	options/format	description	recommended value
sss_data_type	default	constant values defined in the code	
	clim	climatological data	
	ncar	POP ocean forcing data	
sst_data_type	default	constant values defined in the code	
	clim	climatological data	
	ncar	POP ocean forcing data	
trestore	integer	sst restoring time scale (days)	
pointer_file	pointer filename	contains restart filename	
ycycle	integer	number of years in forcing data cycle	
year_init	yyyy	the initial year, if not using restart	

Table 4: Namelist options (continued from previous page).

Index of primary variables and parameters

This index defines many of the symbols used frequently in the ice model code. Values appearing in this list are fixed or recommended; most namelist parameters are indicated (•) with their default values. For other namelist options, see Table 4. All quantities in the code are expressed in MKS units (temperatures may take either Celsius or Kelvin units).

A

advection	• type of advection algorithm used	'remap'
ahmax	thickness above which ice albedo is constant	0.5 m
aice0	fractional open water area	
aice(n)	total concentration of ice in grid cell (in category n)	
aice_init	concentration of ice at beginning of dt (for diagnostics)	
albicei	• near infrared ice albedo for thicker ice	0.36
albicev	• visible ice albedo for thicker ice	0.78
albsnowi	• near infrared, cold snow albedo	0.70
albsnowv	• visible, cold snow albedo	0.98
albocn	ocean albedo	0.06
alpha	floe shape constant for lateral melt	0.66
astar	e-folding scale for participation function	0.05
awtidf	weighting factor for near-ir, diffuse albedo	0.16
awtidr	weighting factor for near-ir, direct albedo	0.31
awtvdf	weighting factor for visible, diffuse albedo	0.24
awtvdr	weighting factor for visible, direct albedo	0.29
ANGLE	for conversions between the POP grid and latitude-longitude grids	
ANGLET	ANGLE converted to T-cells	
atm_data_dir	• directory for atmospheric forcing data	
avgsiz	number of fields that can be written to history file	91

C

Cf	ratio of ridging work to PE change in ridging	17.
char_len	length of character variable strings	80
char_len_long	length of longer character variable strings	128
check_step	time step on which to begin writing debugging data	
cldf	cloud fraction	
congel	basal ice growth	m
cosw	cosine of the turning angle in water	1.
cp_air	specific heat of air	1005.0 J/kg/K
cp_wv	specific heat of water vapor	1.81×10^3 J/kg/K
cp_ice	specific heat of fresh ice	2106. J/kg/K
cp_ocn	specific heat of sea water	4218. J/kg/K
cm_to_m	cm to meters conversion	0.01
c$\langle n \rangle$	real(n)	
Cs	fraction of shear energy contributing to ridging	0.5
Cstar	constant in Hibler ice strength formula	20

D

daidtd	ice area tendency due to dynamics/transport	1/s
daidtt	ice area tendency due to thermodynamics	1/s
dalb_mlt	[see **ice_albedo.F**]	-0.075
dalb_mlti	[see **ice_albedo.F**]	-0.100
dalb_mltv	[see **ice_albedo.F**]	-0.150
dardg1dt	rate of fractional area loss by ridging ice	1/s
dardg2dt	rate of fractional area gain by new ridges	1/s
dvirdgdt	ice volume ridging rate	m/s
dbl_kind	definition of double precision	selected_real_kind(13)
dbug	• write forcing data diagnostics	.false.
Delta	function of strain rates (see Section 3.4)	
depressT	ratio of freezing temperature to salinity of brine	0 .054 deg/psu
diag_file	• diagnostic output file (alternative to standard out)	
diag_type	• where diagnostic output is written	stdout
diagfreq	• how often diagnostic output is written (10 = once per 10 dt)	24
divu	strain rate I component, velocity divergence	1/s
divu_adv	divergence associated with advection	1/s
dragw	drag coefficient for water on ice*ρ_w	0.00536*rhow kg/m^3
dt	• thermodynamics time step	3600. s
dt_dyn	dynamics/transport time step (Δt_{dyn})	
dte	subcycling time step for elastic dynamics (Δt_e)	s
dtei	1/dte, where dte is the EVP subcycling time step	1/s
dT_mlt	[see **ice_albedo.F**]	1. deg
dump_file	• output file for restart dump	
dumpfreq	• dump frequency for restarts, `y`, `m` or `d`	y
dumpfreq_n	• restart output frequency	1
dxt	width of T cell (Δx) through the middle	m
dxu	width of U cell (Δx) through the middle	m

dyt	height of T cell (Δy) through the middle	m
dyu	height of U cell (Δy) through the middle	m
dvidtd	ice volume tendency due to dynamics/transport	m/s
dvidtt	ice volume tendency due to thermodynamics	m/s

E

ecc	yield curve major/minor axis ratio, squared	4.
eice(n)	energy of melting of ice per unit area (in category n)	J/m^2
emissivity	emissivity of snow and ice	0.95
eps04	a small number	10^{-4}
eps11	a small number	10^{-11}
eps12	a small number	10^{-12}
eps13	a small number	10^{-13}
eps15	a small number	10^{-15}
esno(n)	energy of melting of snow per unit area (in category n)	J/m^2
evap	evaporative water flux	kg/m^2 s
evp_damping	• if true, use evp damping procedure [13]	F
eyc	coefficient for calculating the parameter E, 0$<$ eyc $<$1	0.36

F

fcor	Coriolis parameter	1/s
ferrmax	max allowed energy flux error (thermodynamics)	$1.\times10^{-3}$ W/m^2
fhnet	net heat flux to ocean	W/m^2
fhnet_hist	net heat flux to ocean (Fhnet) for history	W/m^2
flat	latent heat flux	W/m^2
floediam	effective floe diameter for lateral melt	300. m
flw	incoming longwave radiation	W/m^2
flwout	outgoing longwave radiation	W/m^2
frain	rainfall rate	kg/m^2/s
frazil	frazil ice growth	m
fresh	fresh water flux to ocean	kg/m^2/s
fresh_hist	fresh water flux (fresh) for history	kg/m^2/s
frzmlt	freezing/melting potential	W/m^2
frz_onset	day of year that freezing begins	
fsalt	net salt flux to ocean	kg/m^2/s
fsalt_hist	salt flux to ocean (fsalt) for history	kg/m^2/s
fsens	sensible heat flux	W/m^2
fsnow	snowfall rate	kg/m^2 s
fsnowrdg	snow fraction that survives in ridging	0.5
fsw	incoming shortwave radiation	W/m^2
fswabs	absorbed shortwave radiation	W/m^2
fswthru	shortwave penetrating to ocean	W/m^2
fswthru_hist	shortwave penetrating to ocean (fswthru) for history	W/m^2
fyear	current data year	
fyear_final	last data year	
fyear_init	• initial data year	

G

gravit	gravitational acceleration	9.80616 m/s^2
grid_file	• input file for grid info	
grid_type	• 'rectangular' or 'displaced_pole' or 'column'	displaced_pole
Gstar	used to compute ridging participation function	0.15

H

hfrazilmin	minimum thickness of new frazil ice	0.05 m
hi_min	minimum ice thickness for thinnest ice category	0.01 m
hicen	ice thickness in category n	m
hin_max	category thickness limits	m
hist_avg	• if true, write averaged data instead of snapshots	T
histfreq	• history output frequency: y, m, w, d or 1	m
history_dir	• path to history output files	
history_file	• history output file prefix	
hm	land/boundary mask, thickness (T-cell)	
hmix	ocean mixed layer depth	20. m
hsnomin	minimum thickness for which T_s is computed	$1.\times10^{-6}$ m
Hstar	determines mean thickness of ridged ice	25. m
HTE	length of eastern edge (Δy) of T-cell	m
HTN	length of northern edge (Δx) of T-cell	m
HTS	length of southern edge (Δx) of T-cell	m
HTW	length of western edge of (Δy) T-cell	m

I

i0vis	fraction of penetrating visible solar radiation	0.70
icells	number of grid cells with specified property (for vectorization)	
ice_ref_salinity	reference salinity for ice-ocean exchanges	4. psu
iceruf	ice surface roughness	$5.\times10^{-4}$ m
icetmask	ice extent mask (T-cell)	
iceumask	ice extent mask (U-cell)	
idate	the date at the end of the current time step (yyyymmdd)	
ierr	general-use error flag	
i(j)hi	last i(j) index of physical domain (local)	
i(j)lo	first i(j) index of physical domain (local)	
ilyr1	index of the top layer in each cat (for eicen)	
ilyrn	index of the bottom layer in each cat (for eicen)	
i(j)mt_global	number of physical gridpoints in x(y) direction, global domain	
i(j)mt_local	total number of gridpoints in x(y) direction, local domain	
int_kind	definition of an integer	kind(1)
ip, jp	local processor coordinates on which to write debugging data	
istep	local step counter for time loop	
istep0	• number of steps taken in previous run	0
istep1	total number of steps at current time step	

K

kappav	visible extinction coefficient in ice, wavelength<700nm	1.4/m
kappan	visible extinction coefficient in ice, wavelength>700nm	17.6/m
kcatbound	• category boundary formula	0
kdyn	• type of dynamics (1 = EVP, 0 = off)	1
kg_to_g	kg to g conversion factor	1000.
kice	thermal conductivity of fresh ice	2.03 W/m/deg
kimin	minimum conductivity of saline ice	W/m/deg
kitd	• type of itd conversions (0 = delta function, 1 = linear remap)	1
kmt_file	• input file for land mask info	
krdg_partic	• ridging participation function	1
krdg_redist	• ridging redistribution function	1
ksno	thermal conductivity of snow	0.30 W/m/deg
kstrength	• ice stength formulation (1= Rothrock 1975, 0= Hibler 1979)	1

L

l_conservation_check	if true, check conservation	
Lfresh	latent heat of melting of fresh ice = Lsub - Lvap	J/kg
lhcoef	transfer coefficient for latent heat	
log_kind	definition of a logical variable	kind(.true.)
Lsub	latent heat of sublimation for fresh water	2.835×10^6 J/kg
Lvap	latent heat of vaporization for fresh water	2.501×10^6 J/kg

M

m_to_cm	meters to cm conversion	100.
m1	constant for lateral melt rate	1.6×10^{-6} m/s deg^{-m2}
m2	constant for lateral melt rate	1.36
m2_to_km2	m^2 to km^2 conversion	1×10^{-6}
mask_n(s)	northern (southern) hemisphere mask	
master_task	task ID for the controlling processor	
mday	day of the month	
meltb	basal ice melt	m
meltl	lateral ice melt	m
meltt	top ice melt	m
melt_onset	day of year that surface melt begins	
month	the month number	
MPI_COMM_ICE	communicator for ice model internal communications (MPI)	
mps_to_cmpdy	m per s to cm per day conversion	8.64×10^6
mps_to_cmpyr	m per s to cm per yr conversion	
mtask	local processor number that writes debugging data	
my_task	task ID for the current processor	

N

nbr_$\langle dir \rangle$	processor numbers for the n, s, e, w neighbor processors	
ncat	number of ice categories	5
ndte	• number of subcycles	120
ndyn_dt	• number of dynamics/advection steps under thermo	1
new_day	flag for beginning new day	
new_month	flag for beginning new month	
new_week	flag for beginning new week	
new_year	flag for beginning new year	
ngroups	number of groups of flux triangles in remapping	5
nilyr	number of ice layers	4
npt	• total number of time steps (dt)	24
ntilay	sum of number of layers in all categories	
ntracer	number of tracers transported in remapping	
nu_diag	unit number for diagnostics output file	6
nu_dump	unit number for dump file for restarting	50
nu_forcing	unit number for forcing data file	49
nu_grid	unit number for grid file	11
nu_kmt	unit number for land mask file	12
nu_nml	unit number for namelist input file	21
nu_restart	unit number for restart input file	50
nu_rst_pointer	unit number for pointer to latest restart file	52
num_ghost_cells	number of rows of ghost cells surrounding each subdomain	1
nyr	year number	

O

oceanmixed_file	• data file containing ocean forcing data	
oceanmixed_ice	• if true, use internal ocean mixed layer	T
ocn_data_dir	• directory for ocean forcing data	
omega	angular velocity of Earth	7.292×10^{-5} rad/s
one	array of ones which is often useful	1.
opening	rate of ice opening due to divergence and shear	1/s

P

p001	1/1000
p01	1/100
p027	1/36
p055	1/18
p1	1/10
p111	1/9
p15	15/100
p166	1/6
p2	1/5
p222	2/9

p25	1/4	
p333	1/3	
p4	2/5	
p5	1/2	
p52083	25/48	
p5625m	-9/16	
p6	3/5	
p666	2/3	
pi	π	
pih	$\pi/2$	
pi2	2π	
pointer_file	• input file for restarting	
potT	atmospheric potential temperature	K
precip_units	• liquid precipitation data units	
print_global	• if true, print global data	F
print_points	• if true, print point data	F
Pstar	ice strength parameter	2.75×10^{4}N/m
puny	a small positive number	1×10^{-11}

Q

Qa	specific humidity at 10 m	kg/kg
qdp	deep ocean heat flux	W/m^2
qqqice	for saturated specific humidity over ice	1.16378×10^{7}kg/m^3
qqqocn	for saturated specific humidity over ocean	6.275724×10^{6}kg/m^3
Qref	2m atmospheric reference specific humidity	kg/kg

R

rad_to_deg	degree-radian conversion	$180/\pi$
radius	earth radius	6.37×10^{6} m
real_kind	definition of single precision real	selected_real_kind(6)
restart	• if true, initialize using restart file instead of defaults	T
restart_dir	• path to restart/dump files	
restore_sst	• restore sst to data	
rhoa	air density	kg/m^3
rhofresh	density of fresh water	1000.0 kg/m^3
rhoi	density of ice	917. kg/m^3
rhos	density of snow	330. kg/m^3
rhow	density of seawater	1026. kg/m^3
rnilyr	real(nlyr)	
rside	fraction of ice that melts laterally	

S

saltmax	max salinity, at ice base	3.2 ppm
sec	seconds elasped into idate	
secday	number of seconds in a day	86400.

shear	strain rate II component	1/s
shcoef	transfer coefficient for sensible heat	
sig1(2)	principal stress components (diagnostic)	
sinw	sine of the turning angle in water	0.
snoice	snow-ice formation	m
snowpatch	length scale for parameterizing nonuniform snow coverage	0.02 m
spval	special value (generally over land or undefined regions, in place of 0)	$10^3 0$
ss_tltx(y)	sea surface slope in the x(y) direction	m/m
sss	sea surface salinity	psu
sss_data_type	• source of surface salinity data	
sst	sea surface temperature	C
sst_data_type	• source of surface temperature data	
stefan-boltzmann	Stefan-Boltzmann constant	5.67×10^{-8} W/m^{2}K^4
stop_now	if 1, end program execution	
strairx(y)	stress on ice by air in the x(y)-direction (centered in U cell)	N/m^2
strairx(y)T	stress on ice by air, x(y)-direction (centered in T cell)	N/m^2
strength	ice strength (pressure)	N/m
stressp	internal ice stress, $\sigma_{11} + \sigma_{22}$	N/m
stressm	internal ice stress, $\sigma_{11} - \sigma_{22}$	N/m
stress12	internal ice stress, σ_{12}	N/m
strintx(y)	divergence of internal ice stress, x(y)	N/m^2
strocnx(y)	ice-ocean stress in the x(y)-direction (U-cell)	N/m^2
strocnx(y)T	ice-ocean stress, x(y)-dir. (T-cell)	N/m^2
strtltx(y)	surface stress due to sea surface slope	N/m^2
swv(n)dr(f)	incoming shortwave radiation, visible (near IR), direct (diffuse)	W/m^2

T

Tair	air temperature at 10 m	K
tarea	area of T-cell	m^2
tarean	area of northern hemisphere T-cells	m^2
tarear	1/tarea	1/m^2
tareas	area of southern hemisphere T-cells	m^2
Tf	freezing temperature	C
Tffresh	freezing temp of fresh ice	273.15 K
time	total elapsed time	s
time_forc	time of last forcing update	s
Timelt	melting temperature of ice top surface	0. C
tinyarea	puny * tarea	m^2
TLAT	latitude of cell center	radians
TLON	longitude of cell center	radians
tmask	land/boundary mask, thickness (T-cell)	
tmass	total mass of ice and snow	kg/m^2
Tmin	minimum allowed internal temperature	-100° C
Tref	2m atmospheric reference temperature	K
trestore	• sst restoring time scale	days
Tsfc(n)	temperature of ice/snow top surface (in category n)	C
Tsf_errmax	max allowed T_{sfc} error (thermodynamics)	$5.\times10^{-4}$deg
Tsmelt	melting temperature of snow top surface	0. C

TTTice	for saturated specific humidity over ice	5897.8 K
TTTocn	for saturated specific humidity over ocean	5107.4 K

U

uarea	area of U-cell	m^2
uarear	1/uarea	
u(v)atm	wind velocity, x(y)	m/s
ULON	longitude of U-cell centers	radians
ULAT	latitude of U-cell centers	radians
umask	land/boundary mask, velocity (U-cell)	
umin	min wind speed for turbulent fluxes	1. m/s
u(v)ocn	ocean current, x(y)-direction	m/s
uvel(vvel)	x(y)-component of velocity	m/s
uvm	land/boundary mask, velocity (U-cell)	

V

vice(n)	volume per unit area of ice (in category n)	m
vonkar	von Karman constant	0.4
vsno(n)	volume per unit area of snow (in category n)	m

W

week	week of the year	
wind	wind speed	m/s
work_g1	allocatable, dbl_kind work array	
work_g2	allocatable, dbl_kind work array	
work_gr	allocatable, real_kind work array	
write_history	if true, write history now	
write_ic	if true, write initial conditions now	
work_l1	(imt_local, jmt_local) work array	
work_l2	(imt_local, jmt_local) work array	
work_a	(ilo:ihi, jlo:jhi) work array	
work_b	(ilo:ihi, jlo:jhi) work array	
write_restart	if 1, write restart now	

Y

ycycle	• number of years in forcing data cycle	
yday	day of the year	
year_init	• the initial year	

Z

zlvl	atmospheric level height	m
zref	reference height for stability	10. m
zTrf	reference height for T_{ref}, Q_{ref}	2. m
zvir	gas constant (water vapor)/gas constant (air) - 1	0.606

Index

advection, *see* transport
albedo, 3, *26*, 35, 38, 43
AOMIP, 44
area, ice, *see* ice fraction

bilinear, 15, 25, 37
boundary
  communication, 38
  condition, *37–38*, 41
  layer, *4–6*, 7
  thickness category, 7, *18–20*

categories, thickness, *see* thickness distribution
CCSM, 2, 4, 38, 42, 43
CFL condition, 9, *39*
column model, 37
Community Climate System Model, *see* CCSM
concentration, *see* ice fraction
conservation, 4, 8, 18, 20, 21, 29, 34, 40
conservation equation, *see* transport
continuity equation, *see* transport
Coriolis, 4, 24
coupling, *see* flux coupler
currents, ocean, 3, 6, 28

damping timescale, 24
density
  atmosphere, 3, 4, 6, 28
  ice or snow, 28, 34
  ocean, 6, 28
diagnostics, 6, 36, 40, *40*, 43
dynamics
  elastic-viscous-plastic, *see* elastic-viscous-plastic
  ridging, *see* ridging
  transport, *see* transport

elastic
  -viscous-plastic dynamics, 2, 8, 22, *23–25*, 36, 38–39, 41
  waves, 23, 24, 39
energy, *see* enthalpy
enthalpy, 7, 9–11, 17, 26, *32–34*
evaporation, 3, *33*
EVP, *see* elastic-viscous-plastic dynamics

flux coupler, 2, *4–7*, 28, 34, 36, 38, 40–43
fraction, ice, *see* ice fraction
frazil, 6
freeboard, 34
freezing potential, 3, *6*
fresh water flux, 3, 4, 6

grid, 3, 12, 25, 34–36, *37–38*, 39, 42

height
  reference, 3–5
  sea surface, 6
history, 3, 36, 41
history files, 39–40, 43
humidity
  reference, 3, 6
  specific, 3–5, 28

ice, *see individual variables*
  fraction, *4*, 7–9, 17–19, 21–24, 43
  fresh, 2
  growth, 7, 18, 20, *32–34*
ice-ocean stress, 3, 4, *6*, 24
internal stress, *23–25*, 37, 40

LANL, 2, 43, 44
latent heat, 3–6, 26, *28*, 33
lateral melt, 28, 34
leads, *see* open water
longwave, *see* radiation, longwave
Los Alamos National Laboratory, *see* LANL
Los Alamos National Security, LLC, 44

masks, 37–38
mechanical distribution, *see* ridging
melt pond, 4
melting potential, 3, *6*, 33
meltwater, 4, 6, 33
mixed layer, 26, 33, 36
momentum equation, 23
monotonicity, 8–11, 14, 15, 18
MPDATA, 8

namelist, 3, 36, *44, 45*
National Center for Atmospheric Research, *see* NCAR
Naval Postgraduate School, 44
NCAR, 2, 4, 38, 42, 44, 46

Oak Ridge National Laboratory, 41, 44
ocean, *6–7*
    currents, *see* currents, ocean
    heat, 3, 6, *28*
    mixed layer, *see* mixed layer
    stress, *see* ice-ocean stress
    surface height, *see* height, sea surface
    surface slope, *see* slope, sea surface
open water, 4, 6–8, *20*, 21, 22

Parallel Ocean Program, *see* POP
parallelization, 34, 36, 38, 41, 42
POP, 2, 6, 37, 39, 46

radiation
    longwave, 3, 4, 26, *27*
    shortwave, 3, 4, 6, *26–27*, 28–30, 43
rain, 3, 4, 6
reference
    height, *see* height, reference
    humidity, *see* humidity, reference
    temperature, *see* temperature, reference
remapping
    incremental, 8, *8–20*, 36, 39
    linear, *see* transport, thickness
replacement pressure, 24
restart, 35, 36, 38, 40, 41, 43
ridging, 2, 7, 8, *21–23*, 36, 41

salinity
    ice, 6, 26, *26*, 29, 32, 33
    ocean, 3, 6
salt, *see* salinity
sensible heat, 3–6, 26, *27*
shortwave, *see* radiation, shortwave
slope, sea surface, 3, 4, 6, 24
snow, 2–4, 6–8, 10, 20, 22, 24, *25–34*, 43
solar, *see* radiation, shortwave
specific humidity, *see* humidity, specific
stability, 4–6, 35, *38–39*
state variables, 3, 4, 7, 17–18, 36
strain rate, 2, 22, 24
strength, 2, 23
stress
    ice-ocean, *see* ice-ocean stress
    principal, 24, 40
    tensor, *see* internal stress
    wind, *see* wind stress
subcycling, 24, 39
sublimation, *see* evaporation
surface height, *see* height, sea surface

temperature, *25–34*, 46
    atmospheric, 3
    freezing, 6
    ice, 7, 26, 33
    ocean, 3, 6
    potential, 3, 4
    reference, 3, 6
    surface, 4, 5, 7, 8, 11, 17
thermodynamics, *25–34*
thickness
    distribution, 2, 7–9, *18–23*, 26, 36, 39
    ice or snow, 6–11, 18, 23, 24, 26, 33–34, 37
    space, *see* transport, thickness
timers, 2, 36, 40, 43
transport, 2, 7–20, 36, 38, 39, 41
    horizontal, 2, *8–18*
    thickness, *18–20*
turbulent fluxes
    latent heat, *see* latent heat
    sensible heat, *see* sensible heat
    wind stress, *see* wind stress

upwind, 8

van Leer, 11, 14
velocity, ice, 2, 5–9, 11, 15, 16, *23–25*, 37
volume, ice or snow, 7, 8, 10, 16–20, 22, 23, 37

water, open, *see* open water
wind
    stress, 3, 4, *5*, 24
    velocity, 3–5, 39

References

[1] T. L. Amundrud, H. Malling, and R. G. Ingram. Geometrical constraints on the evolution of ridged sea ice. *J. Geophys. Res.*, 109, 2004. C06005, doi:10.1029/2003JC002251.

[2] C. M. Bitz, M. M. Holland, A. J. Weaver, and M. Eby. Simulating the ice-thickness distribution in a coupled climate model. *J. Geophys. Res.–Oceans*, 106:2441–2463, 2001.

[3] C. M. Bitz and W. H. Lipscomb. An energy-conserving thermodynamic sea ice model for climate study. *J. Geophys. Res.–Oceans*, 104:15669–15677, 1999.

[4] W. M. Connolley, J. M. Gregory, E. C. Hunke, and A. J. McLaren. On the consistent scaling of terms in the sea ice dynamics equation. *J. Phys. Oceanogr.*, 34:1776–1780, 2004.

[5] J. K. Dukowicz and J. R. Baumgardner. Incremental remapping as a transport/advection algorithm. *J. Comput. Phys.*, 160:318–335, 2000.

[6] J. K. Dukowicz, R. D. Smith, and R. C. Malone. A reformulation and implementation of the Bryan-Cox-Semtner ocean model on the connection machine. *J. Atmos. Oceanic Technol.*, 10:195–208, 1993.

[7] J. K. Dukowicz, R. D. Smith, and R. C. Malone. Implicit free-surface method for the Bryan-Cox-Semtner ocean model. *J. Geophys. Res.–Oceans*, 99:7991–8014, 1994.

[8] E. E. Ebert, J. L. Schramm, and J. A. Curry. Disposition of solar radiation in sea ice and the upper ocean. *J. Geophys. Res.–Oceans*, 100:15,965–15,975, 1995.

[9] G. M. Flato and W. D. Hibler. Ridging and strength in modeling the thickness distribution of Arctic sea ice. *J. Geophys. Res.–Oceans*, 100:18611–18626, 1995.

[10] C. A. Geiger, W. D. Hibler, and S. F. Ackley. Large-scale sea ice drift and deformation: Comparison between models and observations in the western Weddell Sea during 1992. *J. Geophys. Res.–Oceans*, 103:21893–21913, 1998.

[11] W. D. Hibler. A dynamic thermodynamic sea ice model. *J. Phys. Oceanogr.*, 9:817–846, 1979.

[12] W. D. Hibler. Modeling a variable thickness sea ice cover. *Mon. Wea. Rev.*, 108:1943–1973, 1980.

[13] E. C. Hunke. Viscous-plastic sea ice dynamics with the EVP model: Linearization issues. *J. Comput. Phys.*, 170:18–38, 2001.

[14] E. C. Hunke and J. K. Dukowicz. An elastic-viscous-plastic model for sea ice dynamics. *J. Phys. Oceanogr.*, 27:1849–1867, 1997.

[15] E. C. Hunke and J. K. Dukowicz. The Elastic-Viscous-Plastic sea ice dynamics model in general orthogonal curvilinear coordinates on a sphere—Effect of metric terms. *Mon. Wea. Rev.*, 130:1848–1865, 2002.

[16] E. C. Hunke and J. K. Dukowicz. The sea ice momentum equation in the free drift regime. Technical Report LA-UR-03-2219, Los Alamos National Laboratory, 2003.

[17] E. C. Hunke and Y. Zhang. A comparison of sea ice dynamics models at high resolution. *Mon. Wea. Rev.*, 127:396–408, 1999.

[18] R. E. Jordan, E. L. Andreas, and A. P. Makshtas. Heat budget of snow-covered sea ice at North Pole 4. *J. Geophys. Res.–Oceans*, 104:7785–7806, 1999.

[19] B. G. Kauffman and W. G. Large. The CCSM coupler, version 5.0.1. Technical note, National Center for Atmospheric Research, August 2002. http://www.ccsm.ucar.edu/models/.

[20] W. H. Lipscomb. *Modeling the Thickness Distribution of Arctic Sea Ice*. PhD thesis, Dept. of Atmospheric Sciences, Univ. of Washington, Seattle, 1998.

[21] W. H. Lipscomb. Remapping the thickness distribution in sea ice models. *J. Geophys. Res.–Oceans*, 106:13,989–14,000, 2001.

[22] W. H. Lipscomb and E. C. Hunke. Modeling sea ice transport using incremental remapping. *Mon. Wea. Rev.*, 132:1341–1354, 2004.

[23] W. H. Lipscomb, E. C. Hunke, W. Maslowski, and J. Jakacki. Improving ridging schemes for high-resolution sea ice models. *J. Geophys. Res.–Oceans*, 2006. In press.

[24] G. A. Maykut. Large-scale heat exchange and ice production in the central Arctic. *J. Geophys. Res.–Oceans*, 87:7971–7984, 1982.

[25] G. A. Maykut and M. G. McPhee. Solar heating of the Arctic mixed layer. *J. Geophys. Res.–Oceans*, 100:24691–24703, 1995.

[26] G. A. Maykut and D. K. Perovich. The role of shortwave radiation in the summer decay of a sea ice cover. *J. Geophys. Res.*, 92(C7):7032–7044, 1987.

[27] G. A. Maykut and N. Untersteiner. Some results from a time dependent thermodynamic model of sea ice. *J. Geophys. Res.*, 76:1550–1575, 1971.

[28] R. J. Murray. Explicit generation of orthogonal grids for ocean models. *J. Comput. Phys.*, 126:251–273, 1996.

[29] N. Ono. Specific heat and heat of fusion of sea ice. In H. Oura, editor, *Physics of Snow and Ice*, volume 1, pages 599–610. Institute of Low Temperature Science, Hokkaido, Japan, 1967.

[30] D. A. Rothrock. The energetics of the plastic deformation of pack ice by ridging. *J. Geophys. Res.*, 80:4514–4519, 1975.

[31] W. Schwarzacher. Pack ice studies in the Arctic Ocean. *J. Geophys. Res.*, 64:2357–2367, 1959.

[32] R. D. Smith, J. K. Dukowicz, and R. C. Malone. Parallel ocean general circulation modeling. *Physica D*, 60:38–61, 1992.

[33] R. D. Smith, S. Kortas, and B. Meltz. Curvilinear coordinates for global ocean models. Technical Report LA-UR-95-1146, Los Alamos National Laboratory, 1995.

[34] P. K. Smolarkiewicz. A fully multidimensional positive definite advection transport algorithm with small implicit diffusion. *J. Comput. Phys.*, 54:325–362, 1984.

[35] M. Steele. Sea ice melting and floe geometry in a simple ice-ocean model. *J. Geophys. Res.*, 97(C11):17729–17738, 1992.

[36] M. Steele, J. Zhang, D. Rothrock, and H. Stern. The force balance of sea ice in a numerical model of the Arctic Ocean. *J. Geophys. Res.–Oceans*, 102:21061–21079, 1997.

[37] A. H. Stroud. *Approximate Calculation of Multiple Integrals*. Prentice-Hall, Englewood Cliffs, New Jersey, 1971. 431 pp.

[38] A. S. Thorndike, D. A. Rothrock, G. A. Maykut, and R. Colony. The thickness distribution of sea ice. *J. Geophys. Res.*, 80:4501–4513, 1975.

[39] N. Untersteiner. Calculations of temperature regime and heat budget of sea ice in the Central Arctic. *J. Geophys. Res.*, 69:4755–4766, 1964.